OBJECT REPRESENTATION AND RECOGNITION IN MACHINE VISION

Michael Oliver Shneier

Ph.D
University of Edinburgh
1978

Abstract

This thesis is concerned with the representation and recognition of objects by computer. A way of representing objects in terms of primitives, or basic descriptive elements, and relations between primitives is presented and discussed. The representation involves abstracting properties of individual scene elements that serve to describe a class of such scene elements. Similarly, relations that hold for particular instances of objects are generalized to ensure that they will be valid for all instances. By sharing all common primitives and relations, the set of models can be made very compact, and yet preserve many desirable properties. A computer implementation of a version of the representation was developed. It is used to illustrate the descriptive power of the representation.

A recognition algorithm is presented that efficiently uses the representation to recognize real world scenes. The recognition involves two passes over the data. In the first pass each scene element is tentatively interpreted as belonging to some subset of the known models, on the basis of matching scene elements with model primitives and performing relational tests. In the second stage the hypotheses are examined by means of a constraint analysis algorithm that attempts to find the globally best interpretation for each scene element. The value of an interpretation is based on the number of expected relations that were actually found to hold between scene elements for each possible interpretation.

Examples of applying the modelling and recognition system are shown in two domains. The primary domain is that of three-dimensional vision, with data provided by a triangulation ranging device. A secondary domain of word recognition and spelling correction is presented to show the power and versatility of the system.

I composed this thesis myself, and the work presented in it is my own.

Acknowledgements

I should like to thank the innumerable people who provided support and encouragement during the progress of this work. Most especially, I am grateful to Robin Popplestone, my supervisor, and to Pat Ambler, Peter Suzman, Bill Clocksin, Ben du Boulay, and Chris Mellish. I also benefited from discussions with the AI group at Essex, especially with Mike Brady, and with the group at Sussex.

I should like to thank Mrs Hiddlestone, Headmistress of St Margaret's School, Edinburgh, for permission to use the excerpts from the school magazine in Chapter 8.

Parts of Chapter 6 appeared earlier in the paper "Recognition Using Semantic Constraints" presented at the 5th IJCAI, MIT, Cambridge, Mass.

The research reported here was partly supported by a grant from the University of the Witwatersrand Council Postgraduate Scholarship Fund.

CONTENTS

1.0 INTRODUCTION

The way in which information is represented is critically important to the effectiveness of any system in Artificial Intelligence. Using a good representation, tasks that would otherwise be intractable become amenable to solution. In machine vision, representations have usually been tailored to specific, small domains. Recognition algorithms have been developed that depend on specific knowledge about the objects in the domain for their effectiveness. This thesis presents a new representation for describing objects and a recognition algorithm that takes advantage of the form of this representation to recognize objects in a robust and efficient manner. The system is intended to be generally applicable, but was developed primarily for the domain of machine vision. It has also been applied to a second domain, that of spelling correction. Both of these domains will be discussed, but the presentation of the system will be largely in terms of machine vision.

1.1 REPRESENTATION

A representation has been developed that takes advantage of similarities in the descriptions, or models, of objects. If two parts of an object, or two parts of different objects, have the same description, then these parts are represented by a single description in the representation. The basic idea is that of sharing descriptions that are common within models and across models. When this sharing is accompanied by a comprehensive indexing scheme, it leads to a representation that has many advantages over previous representations (e.g. Barrow and Popplestone (1971), Falk (1972), Ambler et al. (1975)) and suffers few disadvantages.

Some constraints were placed on the implementation of the system. The first constraint was that the descriptions of objects should be acquired automatically from visual data. This is a sensible constraint because it allows the domain to be changed easily and confronts the problems of representing objects from the real world, rather than of representing artificially-created objects. The second constraint was that the representation should not be domain-dependent. It should be possible to change or expand the domain with as little effort as possible and the kind of information known about objects should be acquired entirely from the objects themselves rather than being partly dependent on built-in assumptions about the domain. This constraint would be useful in a robot assembly domain, where different assembly tasks might use widely different parts. It would not be practical to re-program the system for each new task. It is preferable to be able to acquire descriptions of the parts visually.

Some of the problems to be confronted when a representation and

a recognition system are being designed become amenable to solution in a more natural way using the new representation than using earlier representations. It becomes easier to compare descriptions of objects, the domain of objects may be changed or expanded with very little effort, and the problems of matching symmetrical objects disappear. There is a great deal of flexibility in the choice of how to describe objects, and the form of the representation makes recognition of objects easier.

The ability to compare descriptions is one of the criteria for a useful representation. In the past, objects have each been described separately and compared by finding correspondences between two structures. In the representation presented here, a single structure is shared by all models, with parts of models that are similar being represented by the same part of the structure. Thus, instead of painfully comparing two structures and finding points of correspondence, the single structure can be examined and similarities and differences between object models can be found immediately. Similarities are indicated by shared structure, and differences are indicated by structure that is not shared.

Because the form of the structure is simple and does not include assumptions about the domain, it becomes easy to change the set of objects that are represented. Objects are described in terms of their parts and relations between their parts. To change the domain it is only necessary to provide a means of extracting descriptions of parts of the objects to be described and relations between them. The form of structure of the representation remains unchanged. This is in contrast to the usual requirement of reprogramming the descriptions or building in a new set of assumptions.

An important requirement for a representation and recognition system, especially one that is to work with man-made objects, is an ability to deal with symmetry. Many objects have some degree of symmetry, and this has led to problems in previous systems. The difficulty arises when a symmetrical object is matched with its model description. It is not clear what parts of the object should be matched with what parts of the model because there is no unique way of matching when several parts have the same description. This problem has either been avoided by not allowing symmetric objects in the domain (Underwood and Coates (1972)), or it has been solved by resorting to finding all possible correspondences between the object and its model and choosing one of them (eg Barrow et al. (1972)). When there is a large amount of symmetry, as in a cube, this method can be very expensive. Using the representation in this thesis, the problem of symmetry disappears. There is a single description for all the similar parts, so that there is no confusion when matching. Symmetric objects can thus be recognized with the same effort as non-symmetric objects.

1.2 RECOGNITION

A recognition algorithm has been developed that uses knowledge obtained from models of objects to analyze scenes. The algorithm has advantages in that it tailors its recognition strategy to the particular scene it is analyzing, it finds all interpretations for parts of the scene in parallel, and deals effectively with symmetric objects. Recognition involves matching descriptions of parts of a scene with parts of the models of objects in the scene. A lesson that has been learned from previous work (e.g. Falk (1972), Grape

(1973)) is that it is important to make use of knowledge about the objects as early as possible in the process of recognition. This is facilitated by using the representation described here to model objects.

Two problems for a recognition algorithm are how to find the models that are relevant to part of a scene, and how to discriminate between conflicting interpretations of parts of a scene. In the past it has been customary to compare parts of the scene with different models successively, looking for a reasonable match. If an interpretation was later found to be invalid, it was necessary to back up to the point at which it was made, and to try another model, thus wasting some of the work that had been done (eg Falk(1972), Ambler et al. (1975), Winston(1975)). The representation in this thesis does not have separate models for each object. Instead of matching with individual models, parts of the scene are matched with the structure containing all the models. When a fragment of the scene matches with part of the model structure, all those models that share that part of the structure become possible interpretations for the scene fragment. Not only that, but the structure of models can indicate how best to discriminate between interpretations by taking into account the different relationships that parts of the scene will be expected to have in different objects. Thus it is possible to index all models that are relevant to a fragment of a scene at once, without any wasted effort. It was also possible to tailor the discrimination tests that distinguish between interpretations according to the models in the structure and the contents of the scene. The ease with which these tasks can be accomplished is an important reason for the success of the recognition algorithm.

1.3 LAYOUT OF THE THESIS

The thesis is organized as follows. In the next chapter there is a discussion of earlier work that is relevant to its aims, and a statement about where the work presented here fits into the field. The importance of using knowledge about the objects in the scene at an early stage is discussed. The use of relational structure representations, especially in systems that construct their own models, is described, and the importance of constraint analysis in scene analysis is discussed.

Chapter 3 consists of an introductory example to give an idea of how the system fits together and how it accomplishes its aims. This chapter illustrates the modelling and recognition systems that were implemented for machine vision. It shows how models are built, and how the representation eases the work of the recognition system.

Chapters 4, 5, and 6 present the system in detail. Chapter 4 is concerned with the representation, its properties, advantages, and disadvantages. A conventional model describes a cube as being composed of six square sides A, B, C, D, E, F, with A perpendicular to B, C, D, E, and B perpendicular to A, C, E, F, etc. Using the representation presented in this chapter, a cube is described as consisting simply of square sides that are perpendicular to each other. There is a single description of what a square side looks like, and a single perpendicularity relation for square sides. Each of these can be instantiated several times to give the full cube description.

More than one object model may share parts of the representation structure. There is a single structure that represents all models. If a parallelepiped has a side with the same description as the side

of a cube, then the model for the parallelepiped shares the part of the structure of models that concerns the description of the side with the model for the cube. It is argued in the chapter that this representation has numerous advantages over earlier representations, especially in the treatment of symmetry and the flexibility with which objects can be represented.

Chapter 5 presents the implementation of the modelling system. This system acquires descriptions of objects automatically, and represents the models using a version of the representation presented in Chapter 4. The chapter is concerned with the practical problems of implementing a system to acquire models automatically. The implementation uses a slightly restricted version of the representation. The ease with which models are acquired and the way in which the domain may be expanded are illustrated.

In Chapter 6 the recognition algorithm and its implementation are presented and discussed. The recognition algorithm is in two parts. The first finds possible interpretations for scene fragments and assigns confidence levels for each interpretation. The second part of the algorithm examines ambiguous interpretations and tries to retain those that have greatest confidence. It makes use of a constraint analysis algorithm to reduce the number of interpretations for scene fragments. The aim is to find those interpretations that give rise to the greatest global confidence. Chapter 6 also illustrates the power of the recognition algorithm, and its ability to handle objects described in different ways simultaneously. The form of the representation is shown to aid the recognition process, and allows the system to analyze scenes from the real world in an efficient manner.

Chapter 7 gives examples of the system in the domain of machine vision. It illustrates the modelling and recognition systems, and shows how a database of objects is constructed. Various scenes are analyzed and the abilities of the recognition system are demonstrated.

Chapter 8 discusses how the approach was applied to the domain of spelling correction and gives examples of the use of the system in that domain. The spelling correction domain was chosen to show the generality of the representation and recognition methods. The performance of the system in this domain is illustrated, and its success serves to underline the utility of the system. Finally, Chapter 9 presents conclusions and discusses possible extensions to the system.

2.0 RELATED WORK

This chapter surveys some of the earlier research that is relevant to the aims of this thesis and places the current work in perspective. There are three main areas to be highlighted. First, there is the importance of using knowledge about objects in the domain to aid in recognition. Second, there is the development of good representations for object models, with special reference to relational structures. The third idea is that of constraint analysis and its use in vision. These areas will be explored and then the current work will be contrasted with that of earlier authors.

2.1 USING KNOWLEDGE

All systems that perform recognition have to use knowledge about the objects in their domains at some stage in the recognition process. There are two problems involved. The first is how to apply the knowledge, and the second is how to represent it. In this section we concentrate on the first problem, how to use the knowledge.

The first system we consider is that of Roberts(1965). Roberts exerted a great deal of influence on later workers. His work is notable for the way in which he indexed his models and the way in which he calculated the positions of objects in a scene.

Roberts was concerned with recognizing objects and finding their three-dimensional positions. The objects in his domain were based on three prototypes - a cube, a wedge, and a hexagonal prism. An object was in the domain if it could be constructed out of the prototypes by scaling them and "glueing" them together. Roberts assumed that his recognition system would operate on pictures that had been reduced to perfect line drawings.

The prototypes in his system were described by the polygons that comprised their surfaces (called approved polygons) and also by the vertices and lines of the object where the polygons met. Polygons were represented by a list of the lines that formed their edges, together with the angles between the lines. This information was used both to recognize objects and to find transformations from the two-dimensional picture to the three-dimensional prototypes.

Recognition involved matching polygons in the scene with those in the models. The process involved a hierarchy of tests. First, a junction in the picture was sought that was surrounded by polygons of the same sort as appeared in the models (approved polygons). If such a junction occurred, it was matched with model vertices to try to find a correspondence such that all the polygons matched. This match also specified a transformation that mapped points in the picture to points in the model. A least squares error analysis was performed to see whether or not the mapping was sufficiently close. If not, the interpretation was discarded, and another match tried.

If no vertex was found that was surrounded by approved polygons, a line was sought that had approved polygons on either side. These polygons were used to index the models, the transformation being calculated as before. If neither of the tests succeeded, the program

looked at individual polygons with extra lines coming from one of their vertices. Finally, if necessary, it looked at individual junctions with three lines coming out of them, and matched these against vertices in the models.

This process can be interpreted in another way. The first step, looking at approved polygons surrounding a junction, can be seen as applying a complex test to part of the picture. This test requires the polygons to be of a particular type and to be related in a particular way. The relationship is usually ternary, since three polygons usually meet at a junction. Similarly, the second step involves a binary (adjacency) relationship between the polygons. The polygons and the way they are related provide an index to the set of models. For complex tests, the index is likely to be unique, but as tests become looser, so the amount of ambiguity increases.

Roberts used knowledge about the objects to index the prototypes and find interpretations for parts of the picture. The knowledge was used before the picture was transformed from two dimensions to three dimensions because scale factors had to be determined from the models. It was also applied as a part of the separation of compound objects into subparts. Compound objects were made up of instances of several of the prototypes. They were broken up by assigning subparts to various models. The kind of knowledge that could be used and the manner in which it was brought to bear, were restricted in Roberts' system by the implementation. For example, tests were tried in a fixed order, so that a model indexed by a test early in the sequence became a more likely interpretation than models indexed by later tests. This sometimes led to incorrect interpretations because a part of the scene, having once been interpreted, could not have its

interpretation changed. What was needed was a way of finding alternative interpretations based on the evidence available locally, and of deciding on the best interpretation based on more global evidence. Given that the models were tested in a fixed order, there should have been some backtrack mechanism to allow mistaken interpretations to be changed when they were found to be inconsistent. A better strategy would have been not to make interpretations more definite than was warranted by the available information. This is the principle of "least commitment" (Marr (1975)).

Falk (1972) developed a scene analysis system based partly on the work of Roberts. His aim was to describe scenes by recognizing the objects in them and their positions. The advance over Roberts' system was that Falk worked with imperfect line drawings produced from photographs of objects in the domain. The domain consisted of nine fixed-size prototypes, represented by collections of the vertices, edges, and surfaces that made up the objects. The representation also included a description of the appearance of each object from every distinguished (topologically identical) viewpoint.

A major difference between Falk's system and that of Roberts is the way knowledge was used to guide the interpretation of the scene. Falk used two kinds of knowledge. First he used general heuristics concerning the relationships between edges in a scene to separate parts of the scene into objects. Only after the grouping process did he use specific knowledge about the objects to find their interpretations.

The heuristics used to separate the scene into parts were based on the junctions of lines in the scene. Guzman (1968) had used

heuristics of a similar nature to separate objects, but he had based them on the regions in the scene instead of the edges. Falk found that using the edges made the heuristics less error-sensitive. The heuristics themselves were based on general knowledge about the domain of parallelepipeds and not on knowledge about the nine prototypes. In using edge information to segment scenes, Falk placed himself on the line of evolution from Guzman to Huffman (1971) and Clowes (1971) and to Waltz (1975). However, at the time that the heuristics were developed, they were not properly understood. As a result, Falk was able to deal only with slight imperfections, and had to "touch-up" his scenes before analyzing them.

When the scene had been separated into objects in Falk's system, another set of heuristics was applied to find the base of each object, and hence the supports of that object. Missing lines and corners were then added to parts of the scene to complete the objects. The additions were again based on local heuristics concerning the general properties of parallelepipeds. The position of each object in three dimensions was then calculated using a generalization of Roberts' techniques.

Up to this stage, all the knowledge used was concerned with the general properties of the objects in the domain. Only then was knowledge about the objects themselves brought to bear to find interpretations for the objects. Falk defined a hierarchy of tests to be applied in a fixed order. The first involved comparing the number of regions and junctions visible in a scene with that expected by the different models. If this test failed to produce a unique identification, the lengths of base edges and the heights of vertical edges were used to refine the interpretations followed by a further,

fixed set of tests. The aim was to select a prototype to yield a unique interpretation for the scene fragments.

When a prototype had been selected, the next stage of the process was to predict what the picture would look like. If the prediction did not compare well enough with the scene, the prototype was rejected and another call was made to the recognizer.

Falk was more justified than Roberts in using a fixed set of tests to recognize objects. He had already split the scene into parts that were presumed not to interact, so he was able to reject an interpretation if it was too much in error, knowing that other parts of the scene would not be affected.

Falk's system can be criticised for the large amount of ad hoc knowledge it used and for the way this knowledge was programmed into the system. It meant that the system was totally domain-dependent. The system required the early stages of object separation to be perfect because it was not possible to go back and change the segmentation later. There was no interaction between general knowledge about the domain and specific knowledge about the objects in the domain.

Grape (1973), on the other hand, used little domain-specific knowledge that was not directly obtained from his models. He was concerned with analyzing imperfect line drawings of simple blocks world objects. He dealt successfully with scenes containing missing and extra lines by actively looking for objects using clues present in the scene. Grape made more use than Roberts or Falk of knowledge about objects. This knowledge was used throughout the processing.

Grape's models consisted of perspectively invariant views of objects. The models were constructed from a file of endpoints of

lines from two-dimensional views of the objects. The user was responsible for ensuring that views were perspectively consistent, and that measurements of parallelism and relative length of lines were within the required tolerances.

The models were based on lines and the vertices at their ends. A set of compound features, that is, descriptions of chain-wise connected vertices was produced. Usually, the compound features uniquely specified the model they were built from, so that discovering what compound features were present in a scene would usually suffice for the analysis of the scene.

Objects were recognized by indexing into the models on the basis of features extracted from the scene. There was no need for a separate initial segmentation like that used by Falk. First, vertices were formed and their connection with other vertices was established. Then sets of connected vertices were matched with the compound features in the models. An advance of this work, over that of Falk and Roberts, was that a match with a compound feature resulted in all models with that feature being accessed, instead of one preferred model.

The features were used in decreasing order of complexity, until the mapping program found a complete object or set of incomplete objects. In the second case, the number of lines in the features was used to make a choice. The ordering of feature tests was analogous to that of Roberts. By using the most complex tests first there was the greatest possibility of a unique match. Only if a unique match was not found did less complex, and thus less constraining tests need to be applied.

Once a complete object had been identified, all the lines that

made up the view of the object in the scene were located and deleted from the scene. The whole process was then reapplied to the remaining lines as if to a new scene. This was a pity because anything learned about a scene in an earlier iteration was forgotten.

Grape showed that knowledge about objects, sensibly applied, could overcome the problems caused by imperfections in a scene. By actively looking for clues concerning the identity of parts of the scene, the imperfections could be ignored, or their analysis delayed until the scene had been simplified.

2.2 RELATIONAL STRUCTURE REPRESENTATION

The previous section discussed the use of knowledge without specifying how the knowledge was acquired or how it was represented. All the systems described so far made use of fairly complex representations. These were hand generated by the programmer for the domain under consideration.

There is another way of acquiring models for recognition. They can be constructed automatically from visual data. Several systems have been developed to accomplish this task. All of them have made use of a representation based on a relational structure.

A relational structure is defined as a set E of elements, together with a set P of properties and a set R of relations over it. Such structures have proved amenable for representing objects and are not too complex to be constructed automatically.

Underwood and Coates (1972) described a system that automatically constructed models and recognized instances of the models when they were viewed later. These authors were interested in constructing models that would enable them to recognize objects from

any viewpoint. There were some very good ideas contained in this work. The models are particularly interesting. There was a single model constructed from all the views of the object, rather than the usual multiple models using distinguished viewpoints. The structure of the models was independent of the orientation or size of the objects that were modelled.

The domain of objects that was studied by Underwood and Coates was very restricted. They required objects to be convex and to have planar surfaces. The most restrictive constraints were that objects should not be symmetric, and that two surfaces of an object should share at most one edge.

Objects were represented by descriptions of their surfaces and by connectivity relationships between the surfaces. A surface was described by a list of its edges in clockwise order, a set of numbers describing the shape of the surface, and connectivity information describing which surfaces were adjacent. The description was independent of how the object was oriented.

Models were constructed from a series of views. The amount of rotation between views was not specified, but was constrained by two factors. At least two surfaces visible in the preceding view had to be present in the next view. In addition, either a new surface had to be seen, or a new connection discovered in each view.

These constraints guided a matching procedure. Each successive surface was matched into a growing structure, there being at least two surfaces guaranteed to match. Any new surfaces and connections were added as they were discovered. The procedure was terminated when all surfaces and all connectivity relations were known.

Recognition involved matching a view of an isolated object with

all the stored models. The matching procedure was the same for learning and for recognition. The best match was found on the basis of connectivity and surface shape. The matching algorithm was essentially a tree search, matching surfaces in the scene against surfaces in the models, one model at a time. Heuristics were used to prune branches in the search tree. These heuristics exploited constraints such as shape consistency. For example, a match would be rejected if it was found that the shape of the surface in the scene was different from that of the surface in the model. Alternatively, a match would be rejected if the connectivity relations would force two different surfaces to be identified. Where there were ambiguities, error terms were calculated for each identification, and the identifications were ranked according to their respective probabilities.

The system was able to describe objects in its domain and recognize views of the objects very successfully. It was, however, very limited. Perfect data were required, and objects could appear only in isolation. It would be difficult to extend the recognition to cope with scenes containing more than one object because the shape descriptions for the surfaces depended on complete surfaces being visible. The kinds of objects that could be dealt with did not comprise a very useful class because of the restrictions on the shape of surfaces and the exclusion of symmetric objects.

The advantages of using a relational structure to represent the objects became apparent when models were constructed. It was possible to decide when the whole of an object had been described because only then would all the connectivity relations be known and the graph structure complete. Underwood and Coates did not take

advantage of the structure during recognition; their recognition algorithm was extremely inefficient.

Relational structure representations for machine vision have been studied extensively at the University of Edinburgh. Barrow and Popplestone(1971) described a system designed to acquire models of irregularly-shaped objects visually and to recognize instances of the objects. The aims of the project were thus similar to those of Underwood and Coates, but the domain of objects and the solutions to the problems were very different. The objects that Barrow and Popplestone's program recognized included a cup, a pair of spectacles, and a hammer. They based their representation on properties of regions and the relations between them. Their concept of learning was to generalize over a number of pictures, finding the information relevant to object identification, and discarding the rest.

Models consisted of typical views of the objects, described in terms of average characteristics derived from a number of training sessions. An object was placed in a pre-specified viewing position and a correspondence between the model and the expected appearance of the picture was supplied. Regions were described by a number of properties, including a measure of compactness and six shape descriptors. Relations included adjacency and the relative size of regions. The system then worked out values of properties of the regions in the picture and relations between them. The mean and standard deviation of each measurement were calculated for the set of training views and stored with the model.

Matching consisted of setting up correspondences between regions of the picture and regions of the models of views of the object.

This was possible because the models were not descriptions of the objects themselves, but of the sensory data. That is, models were two-dimensional, not three-dimensional. A match was successful if the observed measurements in the picture came within three standard deviations of those in the model. An assumption underlying the matching was that the picture was of a single object wholly contained in the field of view.

Although robust and reasonably successful, this system suffered from a number of disadvantages. The training was very much an interactive procedure. The operator had to supply a correspondence between the model and the view, and had to decide how many different views were required for each model. Models were of views rather than of objects, and the performance of the recognizer was dependent on the training sequence. Symmetrical objects, too, caused the recognizer to perform an excessive amount of work due to the number of possible matches.

What the system did achieve, however, was a step towards a principled representation. It showed the utility of relational structures for describing objects in a uniform manner.

The work of Barrow and Popplestone was considerably extended in the Versatile Assembly System (Ambler *et al.*(1975)). This was designed to separate parts automatically, to recognize them, and to assemble them into some predetermined configuration. For example, it was able to assemble a toy car and a toy ship from a heap of parts. The domain was assumed to consist of rigid parts that were light in colour and visually distinguishable.

Models were more structured than in the Barrow and Popplestone program, but still required one model for each distinguished position

(view) of each object. Objects were described hierarchically, based on a predefined tree of entities. For example:

A part has stable states
A stable state has a view
A view has a region
A region has an outline and a hole set
etc.

A structure of this form was constructed for each part. A series of views of a part were needed to adjust the description in the light of variations caused by the orientation of the part and imperfections in the camera system

Entities possessed properties. For example, a region had an area and a compactness. Binary relations could be defined between components of an entity, such as the distance between centroids of segments in an outline.

Recognition involved a hierarchical match starting with the coarse properties such as area and working down the tree of entities refining the initial analysis. At each stage the set of all possible matches between the sense data and the model in question was maintained. At any stage the match could fail because of a discrepancy and another model would have to be tried.

The set of segments that formed the outline of a part formed a relational structure. Each segment had properties of length and curvature, and there were relations such as adjacency defined between segments. A relational structure matching algorithm was defined, which found all correspondences between two relational structures, even when the structures were not identical. A match was considered to be a set of compatible correspondences preserving the properties and relations of each structure. The best matches were those

corresponding to the largest such sets.

This program was a considerable advance on the earlier Barrow and Popplestone work. The learning process was made much easier by replacing the operator-specified correspondence between picture and model with a matching program. The prespecified hierarchy of entities also made it easy to structure the models. The success of the matching algorithm demonstrated the utility of a theoretical analysis of the matching problem. It was possible to work with partial matches in a uniform manner, and to find several possible correspondences in parallel. The hierarchy of entities proved a powerful tool in structuring the domain, although the hierarchy would have to be reprogrammed if the domain of objects were changed. The program would have been more satisfactory if it could have matched more than one model at a time and if models of objects had been used rather than models of views of objects.

Winston(1975) also used a relational structure in his program for learning descriptions from examples. The kinds of concepts his program learned were those in which objects like bricks or wedges took up specified relationships. For example, it learned that an arch consists of two standing bricks with a space between them and a brick or a wedge lying across both blocks. A set of predefined relations could be defined between the elements (objects like BRICKs or WEDGEs) of the description. These elements formed the nodes in a graph, and nodes were linked by arcs labelled with relations that held between the nodes. The structure was hierarchically arranged in a prespecified ordering. For example, it was known that a BRICK was a less general element than an OBJECT.

Both nodes and relations in the model structure were determined

by successive refinement of an initial structure. The initial structure was obtained from an analysis of a hand generated line drawing of a known example of the concept being modelled, such as an arch. Later information given to the system could consist of further instances of the concept, or "near misses". For example, a relation such as SUPPORTED-BY could have been determined between two blocks in the original example. If a later non-example differed from the original example only in the absence of the SUPPORTED-BY relation, then the model structure for the concept would be altered, changing the SUPPORTED-BY label on the arc to a MUST-BE-SUPPORTED-BY label.

The nodes of the structure could be generalized or made more particular by successive examples. For example a node could be generalized from BRICK to OBJECT. These operations depended on the prespecified hierarchy of objects. The system was enhanced by making the relations into nodes in the structure as well, thus enabling them to be operated on in the same way as other entities.

An interesting idea was to describe substructures that occurred commonly in a model by means of TYPICAL-MEMBER nodes. Thus, only one description would suffice for all occurrences of these substructures (such as the legs of a table).

Winston's program was able to learn a number of concepts, such as what a simple arch should look like. His recognition routines were less impressive. There was no attempt to index into models on the basis of parts of the scene. The recognition procedure worked by describing a scene to be analyzed in the same terms as the models. It then applied a hierarchical structure matcher to find correspondences between the scene description and each model description.

Winston remarks that learning in any form depends on good descriptions. His program's descriptions became very complicated, especially the structures created during recognition. It seems that it would be unable to search for instances of some of his concepts in scenes containing spurious lines. It is doubtful that it could analyze scenes from the real world.

The main contribution of Winston's thesis, however, is concerned with concept-learning rather than training a vision system to recognize particular objects. He therefore does not consider the problems of imperfect information and ambiguity. Training sequences of the kind he suggests make sense in only a small number of situations, and seem unsuited to the task of describing single rigid objects because there is only one example of an object, and there are no "near misses".

The systems discussed in this section had a number of features in common. They all represented objects using descriptions of parts of the objects and relations between them. The descriptions were arranged in a graph or network with individual networks for each model. The descriptions of each object were in one-to-one correspondence with the parts of the object, even when some of the descriptions were the same. Because of these features of the representations, the recognition algorithms also had features in common. Each of the systems matched parts of a scene to be recognized with individual models in some sequence and chose the best match as the interpretation for that part of the scene. All the systems, with the exception of Winston's, had difficulty with symmetric objects. Either they could not handle them at all, or they had to find all interpretations that were equivalent up to symmetry

and choose one of them as the final interpretation. Winston overcame the problem of symmetry by means of TYPICAL-MEMBER nodes. These described what a typical part of an object looked like, and would match with all occurrences of the part in the object.

2.3 CONSTRAINT ANALYSIS

The systems discussed above were restricted to finding interpretations based on fairly local evidence. This was not too important a restriction for systems that analyzed scenes containing single objects. In the scenes of Roberts, Falk, and Grape, however, it was possible that assigning an interpretation on the basis of a small fragment of the scene could make it impossible to analyze the rest of the scene correctly. To enable the global context to influence local interpretations, various constraint analysis techniques have been developed. These techniques involve first finding a set of possible interpretations for parts of the scene, usually based on local clues, and then using constraints between parts of the scene to decide which interpretations best describe the scene as a whole.

The first notable use of this technique in vision is to be found in the work of Waltz(1975). This work was concerned with analyzing perfect line drawings of blocks. He extended the work of Huffman(1971) and Clowes(1971) who had shown how local picture evidence, concentrated at junctions, constrained the interpretation of a two-dimensional picture as a three-dimensional scene. Both Huffman and Clowes introduced the notion of labelling parts of the picture to indicate their interpretation in the scene. Waltz extended this work and considerably enhanced it. He was able to show

that improving the descriptive power of the line labelling schemes of Huffman and Clowes was a positive benefit because spurious interpretations became less likely. This ran counter to the belief current at the time that more information (such as shadow lines) made scene analysis harder. Indeed, Orban (1970) developed Guzman-like heuristics to remove shadows from scenes. His program was written to improve the performance of Guzman's (1968) SEE system. Waltz's illustration of the value of extra information was a result of a better understanding of the blocks world domain.

The main interest of Waltz's work as far as this thesis is concerned is in his filtering procedure. The basis of the Waltz algorithm is the exploitation of physical constraints to exclude incompatible line labels from a line drawing.

In general, the lines forming a junction in the line drawing will have several possible interpretations (or labellings) when examined in isolation. However, when one takes into account the constraint that the label assigned to a line cannot be different at different places along the line, one soon discovers that some of the locally possible labellings are inconsistent with a more global interpretation. Thus, two junctions having a common line are required to receive interpretations that give that line the same label.

Waltz's program propagated this constraint through the network of lines and junctions in a line drawing, ensuring that adjacent junctions always had compatible interpretations. This process was found to converge rapidly in most cases, often to a unique consistent interpretation. The more constraints that could be brought to bear, the more likely it was that a solution would be found. Variations of

this method of finding consistent global interpretations on the basis of local information have become a major tool in machine vision.

A refinement of the Waltz procedure is the use of weights for comparing the likelihood of different interpretations. Weighted interpretations were introduced by Yakimovsky and Feldman (1974) in a system for interpreting real scenes. The system was not based on the Waltz algorithm. Instead, they made use of Bayesian decision theory and semantic information to find meaningful interpretations for pictures of outdoor scenes. Semantic information was in the form of relations. For example, the program expected a region labelled SKY to be above a region labelled HILL in the picture.

Yakimovsky and Feldman employed a region-growing algorithm. It merged adjacent regions one at a time across the weakest boundary. Regions had interpretations, and each interpretation constrained those of its neighbouring regions. The interpretation process involved picking the region with the interpretation that currently enjoyed the highest confidence, and assigning this interpretation to the region. The probabilities of each interpretation of neighbouring regions were affected by fixing an interpretation. The constraints between the regions were used to determine the extent to which each neighbouring interpretation was to change. After calculating the new probabilities, the next best region was interpreted, and the process was iterated.

There were two main difficulties with this method. Firstly, decisions were made on the basis of relatively local information, and once made could not be altered. This could lead to errors. Waltz did not have this problem because he did not fix his interpretations until there were no alternatives. He was able to do this because his

interpretations were not probabilistic, but were either possible or impossible.

The second problem was more fundamental. The use of weightings requires some way of assigning initial probabilities and deciding how the relations should change them. Yakimovsky and Feldman relied on intuitive assignments to initialize their system.

The first of these problems was solved by Barrow and Tenenbaum (1976) in their scene analysis system called MSYS. The second was avoided by insisting that the constraints be provided with each scene that was analyzed.

MSYS is a scene analysis system that has been used to interpret manually partitioned scenes, such as an office scene. It is portrayed as a network of processes, representing independent knowledge sources, that communicate with each other. Each process attempts to explain a fragment of the scene in terms of its limited knowledge. The confidence of an explanation is communicated to other processes attempting to explain overlapping fragments, and may cause them to re-evaluate their own hypotheses. These evaluations continue until an equilibrium state is reached. The confidences in the equilibrium state constitute a preference ordering for the alternate interpretations of each fragment. Confidences are used to guide a heuristic search to the best solution.

The requirements for using the system to solve a problem are that a set of possible assignments be given for each unknown, together with associated _a priori_ likelihoods. Also necessary are a set of constraints that determine the _a posteriori_ likelihood of any assignment for a given instantiation of the remaining unknown variables. A solution of the problem is any consistent and complete

instantiation of the variables. The best solution is that which has the greatest combined *a posteriori* likelihood for its instantiated variables. The likelihood is a measure of how well an instantiation satisfies the constraints imposed on it. It is, in general, a non-linear combination of the likelihoods associated with all other interpretations.

The analysis involves an initial relaxation process to find a consistent likelihood estimate for each locally possible interpretation. The estimate is based both on the *a priori* likelihoods, and on those estimated for other semantically constrained interpretations.

Another two relaxation steps are involved at each stage of a tree search to find the optimal set of assignments, starting at the equilibrium state. At each stage, the highest confidence state is restored, and the highest likelihood interpretation of an uninstantiated variable is assigned to a new copy of that state. The relaxation process is applied to find an updated estimate of the likelihoods in the newly instantiated state. For completeness, relaxation is also applied to the original state with the instantiated variable removed as a possibility. Search always continues in the highest scoring state, terminating when this state is also a terminal state.

Problems that were not solved by Barrow and Tenenbaum concern the assignment of initial interpretations to parts of the scene, and the determination of the constraints that operate between interpretations. Barrow and Tenenbaum required that this information be supplied by the operator of the system with the scene. In this thesis a method will be provided of obtaining the required

interpretations and constraints automatically from models of the objects.

2.4 COMPACT MODELS

The approach taken in this thesis differs from earlier work in two major respects. The first difference concerns the nature of models. Those authors who have previously made use of models of objects (e.g. Falk, Barrow and Popplestone, but not Waltz) have given the term "model" certain implicit characteristics. A model has been defined as a description of an object with a one-to-one relationship between parts of the object and parts of the model. For example, the model for a cube consists of a separate description of each of the six sides of the cube, and of relationships between the sides.

There are a number of reasons why this concept of model, while natural, is not the best for machine vision. It means that each model has to be treated as a separate entity because there is no sharing of structure between models or parts of models. Most systems have performed matching separately on each model in turn, with models being rejected one by one until a match was found. It was desired to be able to match with all possible models in parallel, finding the best match in the most efficient manner possible. Grape(1973) attempted to match in this way by compiling a central list of features against which to match. He was not entirely successful, however, because he had to match against each element of the feature list in turn.

Another reason for re-examining the notion of model concerns the analysis of models of symmetric objects. Underwood and Coates were unable to deal with symmetric objects because of problems of finding

the correct correspondence between parts of the scene and parts of the model. Other authors have had to content themselves with finding all possible correspondences between the scene and a model, and then choosing one of these arbitrarily (e.g. Barrow and Popplestone (1971), Ambler et al. (1975)). The problem arises because the model of a symmetrical object contains several parts that are described in a very similar fashion. When an object model is being constructed from several views, it is not clear when to contruct a new part in the model and when to assume that all parts have already been seen. Similarly, when trying to recognize an object, it is not clear which of a number of parts of a model should be matched against a scene fragment. If the system is to work in a world of man-made objects, it seems advisable to be able to handle symmetry in a more acceptable manner.

The solution to the problems with models adopted in this thesis is to broaden the notion of what it means to be a model. A model is a collection of descriptions of the kind of parts that make up an object and the of the relations between them. There is no requirement of a one-to-one correspondence between object parts and model parts. Rather, there is one model part for each distinct object part. This is similar to systems of mathematical logic, where a model is a collection of axioms describing a world, and where no axioms are repeated.

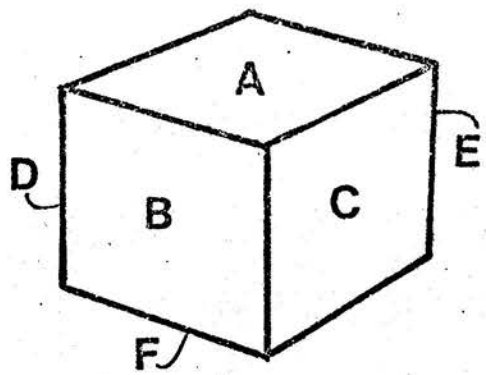
So, for example, a model of a cube consists of only one part description - that of a square plane. Relations are defined between instances of the part, so that a form of relational structure is the basis of the representation.

Figure 2.1 depicts the difference between the conventional

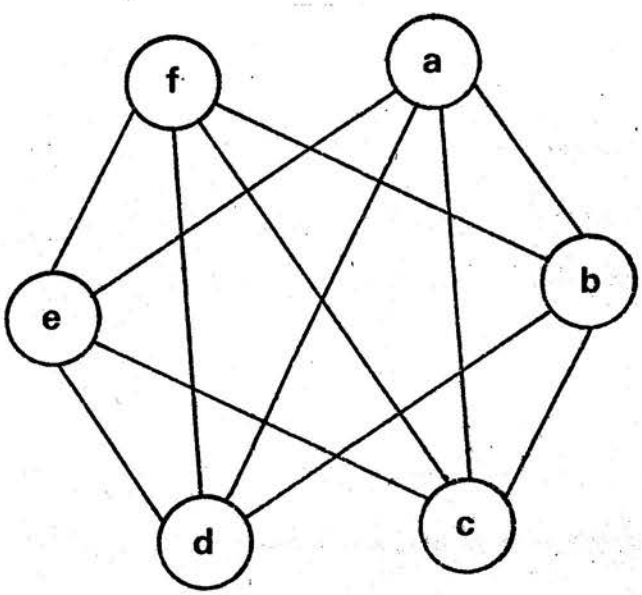
relational structure model of a cube and the model to be used in this thesis. The conventional model has one node for each surface in the cube, whereas the new model has a single node describing all the surfaces. Similarly, while there are many relations defined between the surfaces in the conventional model, the new model needs only a single relation to represent them all. There is a great saving in the size of the model.

In fact, the models are even more like those of the logician than has been described so far. A logician describes a whole world with one set of axioms. Similarly, the models in this representation all share one structure, instead of the usual separation of models of different objects. The advantages of such a system are manifold. Matching is enormously simplified and matches occur with all possible models at once. Models are compact and are easily compared. Similarities are evident in shared structure, and differences are reflected by structure that is not shared.

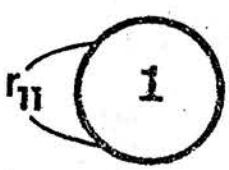
The second difference between the new system and earlier work is more specific, and arises as a result of the formulation of the models as a relational structure. A relational structure is defined as a set E of elements, together with a set of properties P and a set of relations R over it (Ambler et al (1975)). Scene analysis based on relational structures has been concerned with the following kind of matching: Given two relational structures $\langle E_1, P, R\rangle$ and $\langle E_2, P, R\rangle$, define a match between them as a set $T_1 \subseteq E_1$, a set $T_2 \subseteq E_2$, and an isomorphism $\approx$ between T_1 and T_2 preserving properties and relations. That is, $e_1 \approx e_2$ implies $p(e_1)$ iff $p(e_2)$ for each



"cube"



Conventional Model



New Model

Figure 2.1

$p \in P$. Also, $e_1 \approx e_1'$ and $e_2 \approx e_2'$ implies $r(e_1, e_1')$ iff $r(e_2, e_2')$ for each $r \in R$. A match represents a common substructure in the relational structures.

This definition will be changed to allow the two relational structures to have a different relationship. When the recognition problem is described, it will be seen that the structure of models and the structure obtained from the scene being recognized are described in terms of different basic elements. That is, the structure obtained from the scene (called the scene graph) represents the parts of the scene as they actually appear, whereas the structure of models (the graph of models) represents a meta-description of the objects in the scene. Several parts of the scene graph may map into a single node of the graph of models. There is an isomorphism from the scene graph to a structure derived from the graph of models, but not to the graph of models itself. The structure derived from the graph of models is a conventional relational structure with each part of the object described individually, even if the parts have similar descriptions. Relations between the parts are also made explicit.

Thus it is necessary to distinguish three levels in the representation. There is the representation of the actual sense data in the scene graph, there is the structure representing the models of the objects known to the system, and in between there is a structure instantiated from the graph of models to aid in the analysis of the particular scene being studied. The relationship described above for matching relational structures holds between the scene graph and the intermediate structure, but this intermediate structure is only one of many possible such structures that can be instantiated from the graph of models.

Winston, with his TYPICAL-MEMBER labels, went some way towards this position. He did not explore the potential of such a representation, but relied on the conventional structure representation.

The rest of this thesis will show that the position taken on models and relational structure is tenable, and that it has advantages over the conventional approach. Many of the techniques common in vision, such as constraint analysis and knowledge-based interpretation, can be carried over without significant alteration.

3.0 EXAMPLE

Before discussing the system in detail, it is worthwhile looking at an example of the way the implemented version learns about objects and recognizes scenes.

Sense data is acquired by the system by means of a triangulation ranging device (Popplestone and Ambler (1977)). This device is described in Chapter 5. The data available after using the device take the form of a depth map of a scene, for example, the scene in Figure 3.2. Information obtainable from the depth map includes the three-dimensional position of all visible points and the way the points are connected into lines. This information is used to find the surfaces in a scene.

3.1 MODELLING

An object for which a model is to be constructed is placed on a turntable as in Figure 3.1. It is scanned by the ranger, rotated on the turntable by a known amount, and scanned again. This process may be repeated as often as is required, usually until the whole object has been viewed. A range map of the whole object is produced from a composite of all the views. This map gives the three-dimensional positions of all the points visible in any of the scans.

A surface-finding program is applied to this data to find the

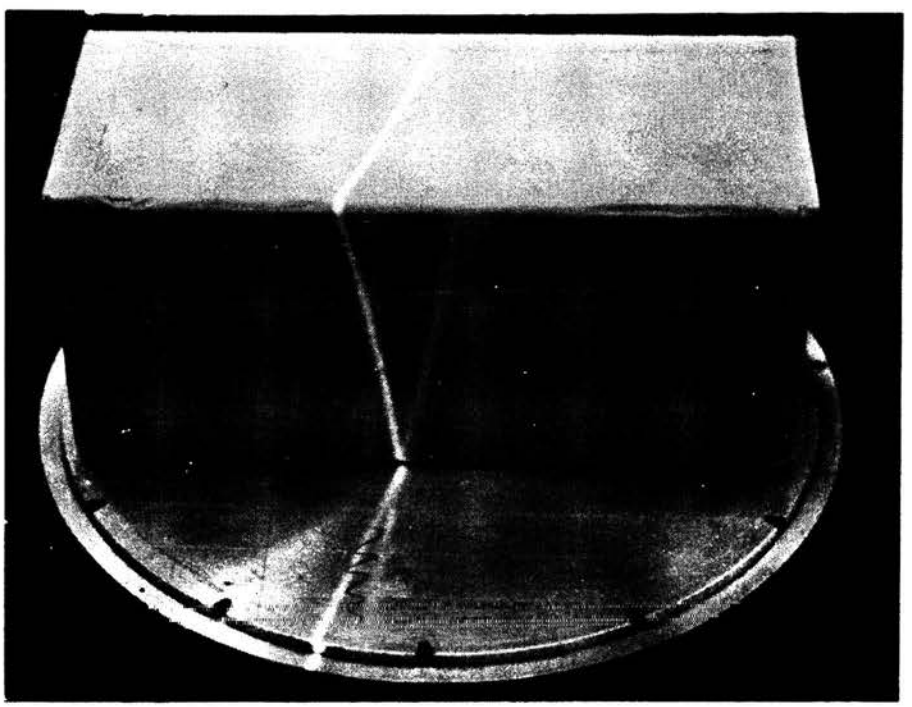


Figure 3.1

planar and cylindrical surfaces. The surfaces are what form the primary input to the modelling system.

Suppose the object of Figure 3.1 has been scanned from four viewpoints, each perpendicular to its predecessor, the first view being the one shown in the Figure. The data from the scans are passed to the surface-finder, and the set of surfaces that make up the object is extracted. The modelling program is applied to these surfaces. It requires three items of input. Firstly, it needs a name for the object. This is supplied by the user. The object will be called MEDBOX for medium-sized box. Secondly, it needs a file containing the data acquired from the scans. Finally, it requires that the system be told the terms in which it is to describe the object. The representation is a relational structure or graph whose nodes are descriptions of surfaces (called PRIMITIVES) derived from the set of scans. The relations between these surfaces, which form the arcs in the graph, are not deduced from the scans, nor are the same set of relations applied to form all models as is more usual. Instead, the user lists a number of functions to be applied to the surfaces in the scan to form relations. The motivation for this is that different objects are best described in different ways, and it was desired not to impose a rigid descriptive formalism on all objects. The call to the modeller is thus:

```
MODEL("MEDBOX",[MEDBOX],[ADJACENT]);
```

ADJACENT will construct an arc in the graph between two surface descriptions if surfaces in the object that fit that description are next to each other.

The first thing that the program does is to work out descriptions of each surface. A description consists only of the

surface type (plane or cylinder) and two numbers denoting the shape of the surface. For rectangular surfaces, these numbers correspond to the length and breadth of the surface. The system has to work out in how many different scans each surface appeared so that it can produce the correct description. For MEDBOX, the program finds the following dimensions (in metres).

```
SURFACE  S1
NEW VIEW  S1        Surface S1 corresponds to the top of MEDBOX. When
NEW VIEW  S1        it is finding the dimensions of the surfaces, the
NEW VIEW  S1        system detects that S1 appeared in all four scans.
 0.1972   0.1155
SURFACE  S2
 0.2235  0.0981
SURFACE  S3
 0.2246  0.0949
SURFACE  S4
 0.1056  0.0942
SURFACE  S5
 0.0964  0.0914
```

The next task of the program is to construct a relational structure to describe the object. First it examines the set of descriptions of surfaces it has formed. If two descriptions are the same, they are both represented by the same node (primitive) in the graph of models. In the example, the two short sides (Surface S4 and Surface S5) are both described by the same node, P2, and the long rectangular surfaces (S1, S2, and S3) are described by another node, P1, making a total of two nodes to describe all the surfaces:

```
PLANE  P1
BODYLIST MEDBOX
EXTENT  0.2175      0.1009

PLANE  P2
BODYLIST MEDBOX
EXTENT  0.101      0.0928
```

Next, the program must relate the nodes by means of the specified relations. To discover whether or not to construct an arc between two nodes, the set of functions defined by the user is

applied to each pair of surfaces. For example, if two surfaces in MEDBOX are found to be next to each other, the function ADJACENT will cause an arc to join the nodes describing the two surfaces. The node may be reflexive if both surfaces are described by the same PRIMITIVE node. The arcs are labelled with the relation of adjacency.

When a more normal relational structure is constructed, such as that of Barrow and Popplestone (1971), a node is created for each scene element, and relations defined between the elements cause arcs to be constructed to link the nodes. In the present structure, however, a node corresponds to a class of scene elements. Therefore, an arc will join two nodes if any member of the first node class is related to any member of the second node class. The arcs are labelled by the relation that caused them to be constructed (or the set of relations if there are several of them). The relations themselves are generalized to schemata. That is, if two instances of each of the nodes joined by an arc are related in the same way, only one relation schema will be constructed to represent both relations.

In the case of MEDBOX, the top and front surfaces of Figure 3.1 are adjacent, and this gives rise to a schema of the form <ADJACENT P1 P1 1>. This says that two instances of the surface described by node P1 are adjacent. It is equally valid for the top and rear surfaces of the block. Similarly the side and front, side and top, side and rear, etc adjacency relations for both small sides of the block are all represented by one schema: <ADJACENT P1 P2 1>.

Thus, the relation schemata needed to describe MEDBOX are:

```
ADJACENT  R2  ARGUMENTS  P1   P2
EXPECTED VALUE  1.0
BODYLIST MEDBOX

ADJACENT  R1  ARGUMENTS  P1   P1
EXPECTED VALUE  1.0
BODYLIST MEDBOX
```

These surface descriptions and relation schemata, together with its name, are all the information necessary to describe the block.

The situation is actually more complex than has been illustrated here. Suppose that some other object had been modelled, and that some of the sides of that object had the same description as some of the sides of MEDBOX. In that case, no new node would be added to the structure, but the node already in existence would be shared with the new object. This sharing makes it necessary to tag the relation schemata on arc joining nodes with the names of the models for which they are valid.

Even with only a single object in the structure, the saving in the size of the model is significant. Only two nodes and two relation schemata are needed to represent the object instead of the usual six nodes and twelve relations. Any symmetric object will gain some benefit from the representation. Most objects that are man-made have some degree of symmetry, so the savings can be significant.

3.2 RECOGNITION

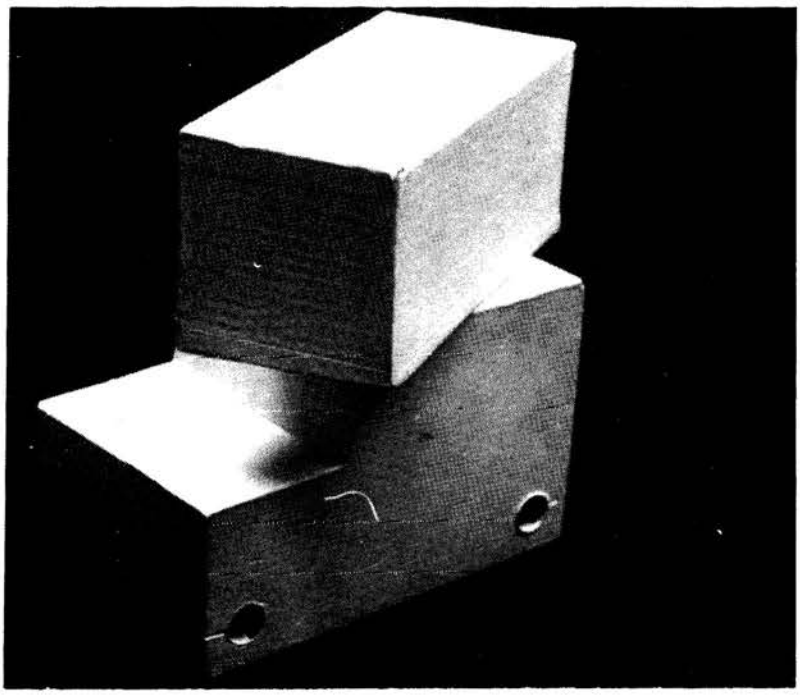
Two problems for a recognition system are first, to find interpretations for the fragments of a scene and second, to find the best interpretation of the scene as a whole. These goals are sometimes in conflict. It might be possible to find an interpretation for a scene fragment that is locally best. This interpretation might, however, make it impossible to find

interpretations for other parts of the scene. The solution to these problems is to perform the recognition in two stages. The first stage finds alternative interpretations for each part of the scene, together with a measure of confidence in each interpretation. The second stage examines the interpretations and tries to find those that give the best result for the whole scene. The recognition algorithm used here has advantages over earlier systems in the way in which it finds interpretations for scene fragments, and for the way in which it discriminates between interpretations. Both these advantages are a consequence of the compact representation.

The first steps of the recognition procedure are similar to those for modelling. For recognition it is assumed that some set of objects has been modelled, and that instances of these objects appear in the scene. The aim is to find which parts of the scene correspond to which objects.

The database of objects for this example consists of a toy car (CAR), a pyramid (PYRAMID), a cylinder (CYL), and three boxes (SMALLBX, MEDBOX, and BIGBOX). Each box has a surface whose dimensions are the same as those of surfaces in each of the other boxes. The full database is given in Chapter 7 (Figure 7.3). Figure 3.3 shows the graph of models for all the objects.

The scene depicted in Figure 3.2a shows the toy car (CAR) with the medium box (MEDBOX) on top of it. Figure 3.2b shows the range map, with a large part of the scene not visible to the ranging device. This map is processed to find the surfaces (labelled 1 to 5 in Figure 3.2b). The surfaces are described in the same way as for modelling, in terms of their type (planar in this case) and their dimensions. The dimensions are as follows.



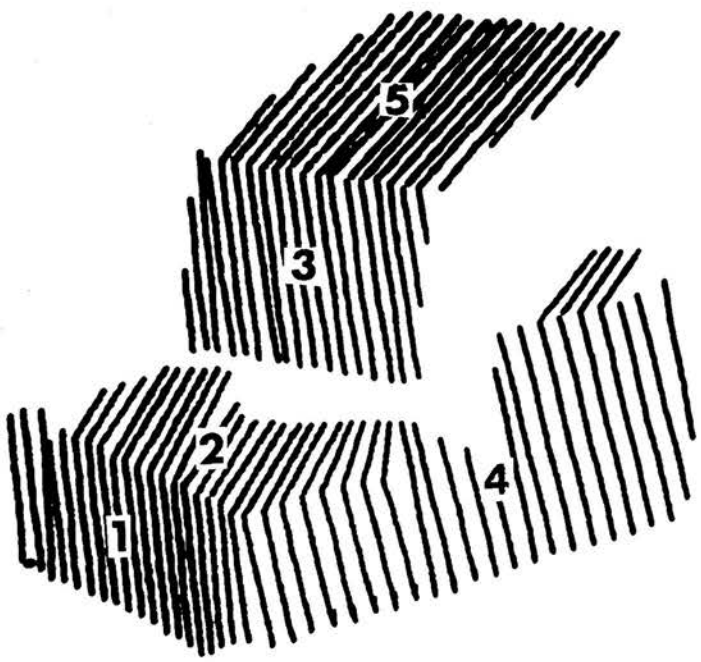


Figure 3.2

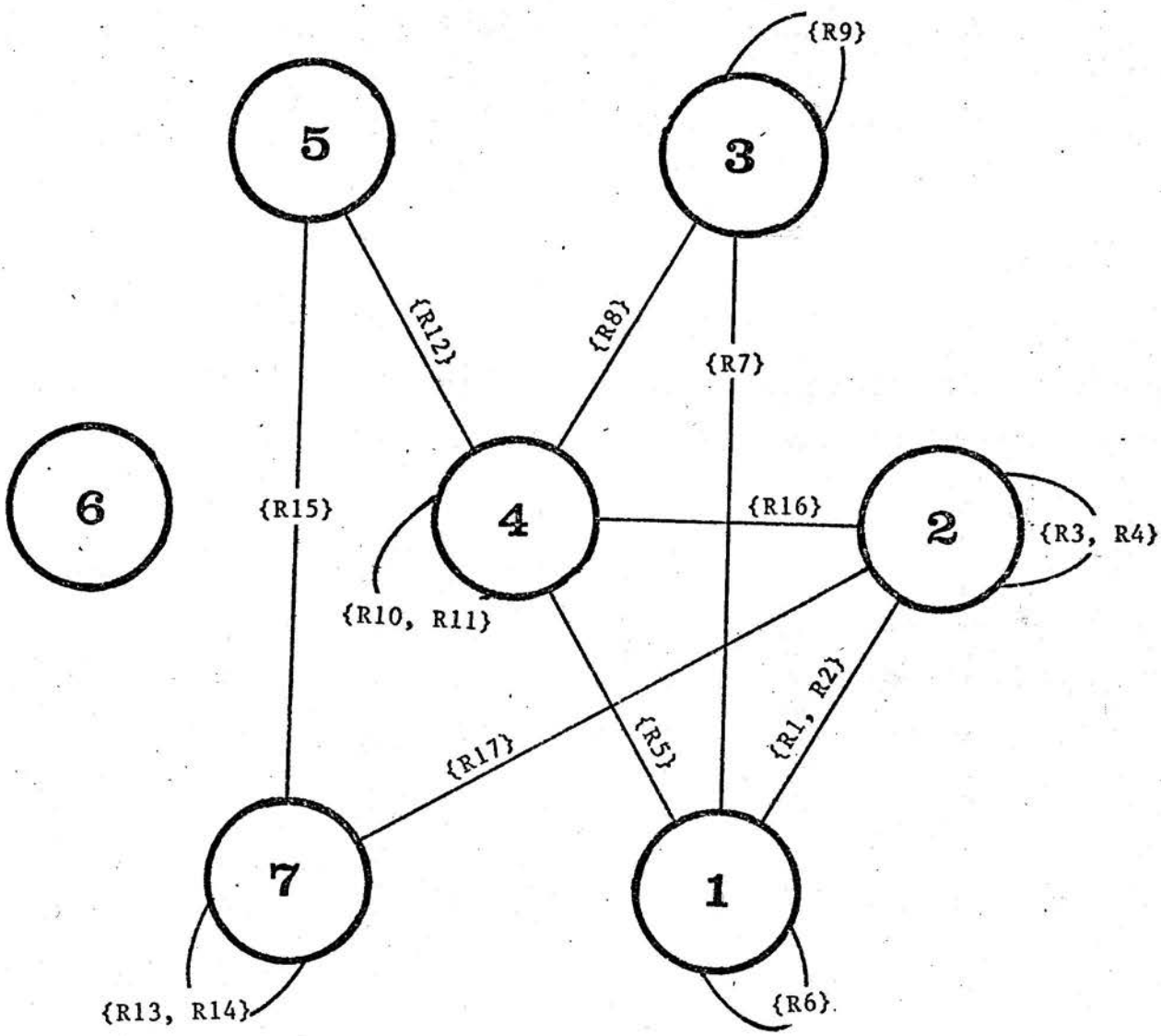


Figure 3.3

```
RECOG([SCENE5]);
SURFACE  S1
 0.131   0.0827
SURFACE  S2
 0.124   0.0782
SURFACE  S3
 0.0986   0.0944
SURFACE  S4
 0.2107   0.1025
SURFACE  S5
 0.2037   0.0552
```

Recognition involves mapping from a structure obtained from the scene (the scene graph) to a structure derived from the graph of models (the intermediate structure). This is a complex procedure that will be described in detail in chapter 6. Here only the particular case of Figure 3.2 is discussed.

The intermediate structure is not constructed explicitly since it is to be isomorphic with the scene graph anyway. Instead information from the scene and from the graph of models is used to construct the scene graph and perform the matching dynamically during the course of the recognition procedure.

The recognition process has two parts. First a structure is constructed that is larger than the final scene graph. This structure contains all possible interpretations of parts of the scene as parts of objects in the graph of models, together with reasons for making each interpretation. Secondly, the initial structure is pruned by means of a constraint analysis algorithm to find those interpretations that have the best supporting reasons.

The process starts by indexing the graph of models using descriptions derived from the surfaces in the scene. A surface description matches with a node in the graph of models if it is of the same type and if its dimensions are sufficiently similar. In practice, a surface description may match with several nodes (e.g.

SURFACE S5 below). A degree of confidence is associated with each match. In the example:

```
SURFACE S1 MATCHED WITH PRIMITIVE P3 WITH CONFIDENCE 5
SURFACE S2 MATCHED WITH PRIMITIVE P3 WITH CONFIDENCE 5
SURFACE S3 MATCHED WITH PRIMITIVE P5 WITH CONFIDENCE 5
SURFACE S4 MATCHED WITH PRIMITIVE P4 WITH CONFIDENCE 5
SURFACE S5 MATCHED WITH PRIMITIVE P4 WITH CONFIDENCE 2
SURFACE S5 MATCHED WITH PRIMITIVE P2 WITH CONFIDENCE 2
```

These matches act as hooks into the graph of models. Each surface in the scene is paired with all the primitives in the graph of models with which it matched. The structure the program builds to describe the scene has these primitive/surface pairs as its elementary nodes.

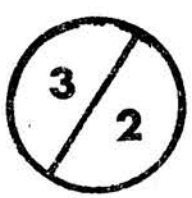
The hooks into the graph of models are used to extract information to guide the construction of the scene graph. The information required concerns the interpretations of each primitive/surface pair in the graph, and the relations between the pairs.

The usual way of finding interpretations for scene fragments is to apply a set of prespecified tests to the scene and use the results to identify a model to match the fragments. The method used here is different. Properties of the parts of the scene are used to index a set of alternative models that might match those parts. When the models are accessed, a set of tests is also made available. Some of these tests are applied, the choice of which particular tests depending on the scene. The tests have two purposes. The first is to decide whether or not an interpretation is possible for a fragment. The second purpose is to supply reasons, or confidences, for the interpretations that are possible. Each interpretation results in a subgraph of the scene graph being created for that interpretation. The aim is to assign each pair to one or more

subgraphs of the scene graph. Each such assignment to a subgraph will require a reason. Reasons are relations that link one pair in a subgraph with other pairs in the same subgraph. Subgraphs correspond to instances of individual models - that is, to the objects in the scene.

In the example, the system starts the process by looking at pair P3/S2 - That is, surface S2 matched with primitive P3. Now, each primitive node in the graph of models has associated with it the set of models for which it is valid. Thus P3 is known to index only the CAR model. In the scene graph, two things happen. Firstly, a distinguished subgraph is defined, called CAR, and secondly the pair P3/S2 is assigned as a node in this subgraph. There are no reasons for this assignment because it is the first to be made to the subgraph.

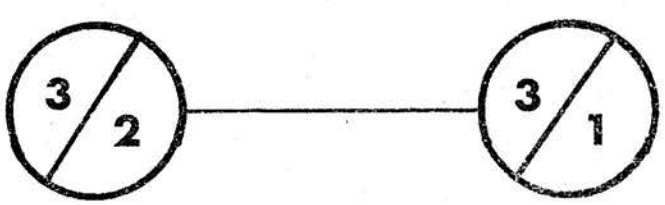
CAR



The program next chooses to assign the pair P3/S1, which corresponds to the front of the car (Figure 3.2). Once again, primitive P3 is valid only for the CAR model, and so P3/S1 must be assigned to an instance of this model. However, it is not known that the scene depicts only one car, and the program insists that there be evidence to support such an assumption. This evidence takes the form of relation schemata in the graph of models. Each arc in the graph of models is labelled with a set of relation schemata. These schemata

may be relevant to different models. A set of schemata may be used to test for compatibility between primitive/surface pairs in a particular subgraph of the scene graph. If a particular relation might be expected to hold between two surfaces that are part of an object, then that relation should be tested. In the case of testing P3/S2 and P3/S1, there is only one such test, that of adjacency. When an object is modelled, the relation schemata are constructed out of functions, primitives, and values. A schema of the form <ADJACENT P3 P3 1> says that if two instances of primitive P3 (i.e. two surfaces like S1 and S2) have the function ADJACENT applied to them, the answer 1 (TRUE) will confirm their compatibility. Therefore, applying ADJACENT to S1 and S2 and comparing the result with that expected by the schema is a way of deciding whether P3/S1 may be assigned to the same instance of CAR as P3/S2 in the scene graph. The test succeeds, so P3/S1 is added to the the subgraph by adding an arc between P3/S1 and P3/S2 in the scene graph. The arc is labelled with the instantiated relation schema <ADJACENT S1 S2 1> that confirmed the compatibility.

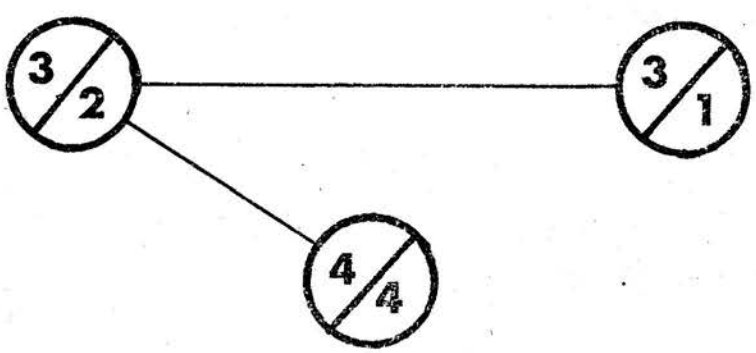
CAR



The next pair to be assigned is P4/S4, representing the side of the car. P4 indexes three models, those for CAR, MEDBOX, and BIGBOX,

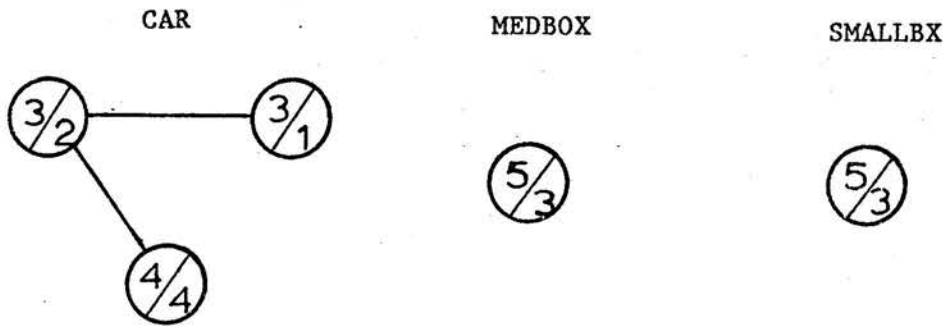
and P4/S4 must be assigned to a subgraph in the scene graph associated with one of these models. At this stage there are two pairs already in the scene graph associated with CAR, and to interpret the new pair P4/S4 as part of the same subgraph, it must be shown that P4/S4 is compatible with at least one of the pairs already in the subgraph (P3/S1 and P3/S2). In fact the test between P3/S1 and P4/S4 fails, although S1 and S4 are actually adjacent. The other test, between P3/S2 and P4/S4 is successful, and is considered sufficient for compatibility. P4/S4 is added to the subgraph of the scene graph, and linked with P3/S2.

CAR



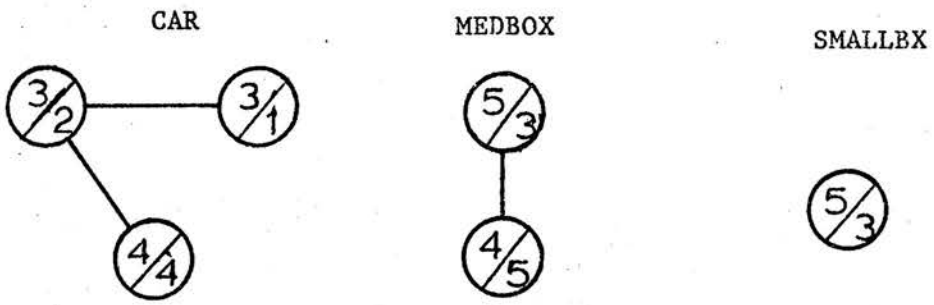
The next pair to be assigned is P5/S3, corresponding to the front of the box. There is no arc in the graph of models between primitive P5 and any of the primitives already associated with surfaces in the scene graph (ie P3 and P4). Therefore there are no tests that can be applied to make P5/S3 compatible with the interpretation CAR. At this stage there are no other subgraphs of the scene graph, so some new interpretation must be found for P5/S3. The system has no way of knowing which of the possible interpretations available for P5/S3 is correct, so assigns subgraphs for each possible interpretation (i.e. for instances of each model

indexed by P5). There are two of these: MEDBOX, which we know to be correct, and SMALLBX. A node is created in each of these subgraphs to correspond to P5/S3.



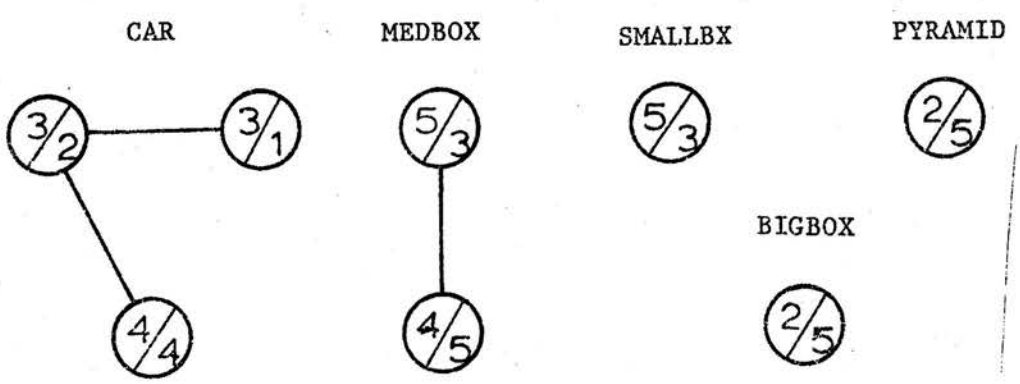
The next pair chosen is P4/S5. There is an arc between P4 and P5 in the graph of models, so that it is possible to test the new pair for compatibility with P5/S3. Applying the adjacency test on the arc proves successful. P5 indexes both MEDBOX and SMALLBX. However, the adjacency relation on the arc between P5 and P4 is valid only for MEDBOX. Therefore, P4/S5 is compatible with the subgraph in the scene graph associated with MEDBOX, but not with that associated with SMALLBX. An arc joins the two pairs in the subgraph for MEDBOX, but P4/S5 is not assigned to the SMALLBX subgraph.

Two other relation tests are applied to find whether or not P4/S5 is compatible with P3/S1 and P3/S2. Both these tests fail, so no further assignments of P4/S5 are made.



At this stage all the surfaces in the scene have been treated. Recall, however, that surface S5 matched with two primitive nodes in the graph of models, forming two pairs to be assigned. It is not yet known which was the better match. The pair P2/S5 must still be assigned.

None of the three existing subgraphs can accept P2/S5 because P2 has no links in the graph of models to any of P3, P4, or P5. The system is forced to assign P2/S5 to all objects indexed by P2, adding two more subgraphs to the scene graph.



The scene graph is now completely built, and consists of five subgraphs. The subgraph whose interpretation is CAR has the three pairs P4/S4, P3/S2, and P3/S1 as nodes. The subgraph associated with

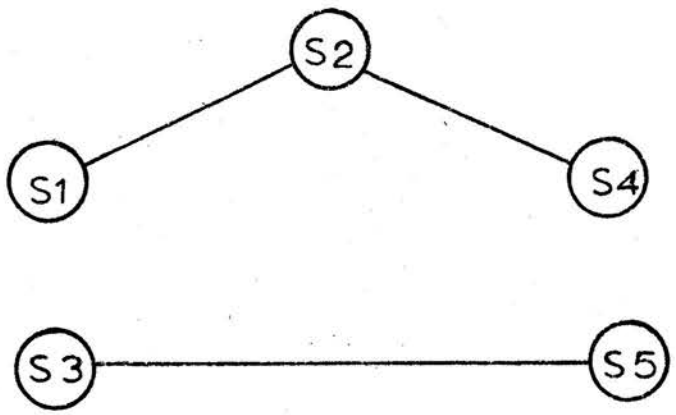
MEDBOX has the two pairs P5/S3 and P4/S5 as its nodes. The other three subgraphs each have one pair assigned to them. The final part of the recognition involves examining a scene graph and deleting surfaces that are insufficiently substantiated. A constraint analysis algorithm is applied to test the relative strengths of assignments of the same surface to several subgraphs. Those assignments that are not supported strongly enough are deleted. A surface may have been matched with more than one primitive, so the algorithm cannot be based on primitive/surface pairs alone.

In the example, surface S3 (pair P5/S3) was assigned to two subgraphs - one for MEDBOX and one for SMALLBX. The system examines these assignments and finds that P5/S3 is justified as a node in the MEDBOX subgraph by its ADJACENT link to P2/S5. In SMALLBX, however, P5/S3 has no justification, being the only pair assigned to SMALLBX. P5/S3 is therefore removed from SMALLBX, and the subgraph is thus deleted from the scene graph.

The other case of a surface being assigned to more than one subgraph arose because surface S5 matched with two primitives, forming the pairs P4/S5 and P2/S5. The nodes in the scene graph corresponding to these pairs are linked through their common surface. The algorithm attempts to find a single interpretation for that surface by deleting as many nodes as it can.

At first there are three subgraphs in which the surface has an interpretation - those of MEDBOX, PYRAMID, and BIGBOX. The pair P4/S5 has a link that justifies its presence in MEDBOX, while the assignments of P2/S5 to PYRAMID and BIGBOX have no such justification. The assignment to MEDBOX is thus retained, and the others deleted. This gives the final interpretation below.

```
CAR
S4
ADJACENT   S4   S2
S1
ADJACENT   S1   S2
S2
ADJACENT   S1   S2
ADJACENT   S4   S2

MEDBOX
S3
ADJACENT   S5   S3
S5
ADJACENT   S5   S3
```



The constraint analysis is not always as simple as it appears in this example. When rival interpretations each have justifying reasons, then more analysis is needed. This analysis takes the form of a kind of filtering. When a node linked to other nodes is deleted, the arcs joining it to those nodes, together with the relations that label the arcs are deleted as well. This means that the justification for the other nodes is weakened, making them more likely to be deleted later. The process of deleting nodes continues until each surface is uniquely interpreted or until all interpretations of a surface are equally justified, in which case the result is ambiguous.

4.0 REPRESENTING MODELS OF OBJECTS

4.1 INTRODUCTION.

This chapter is concerned with the representation of object models for use in the visual domain. The choice of representation has two facets - what is to be represented, and how it is to be represented. That is, what sorts of objects are going to be represented, what features of these objects are considered important or useful, and how features and properties of objects are to be related to each other, and to those of other objects.

A fairly strong constraint will be imposed on the above questions, that of requiring the object models to be learned from examples presented to the standard visual input system. The advantage of having models learned in this way is the ease with which the domain can be changed or expanded. The scope of the modelling system is also a consideration. It is desired to be able to represent objects from a variety of domains with as little re-programming as possible. This chapter deals with the representation in a theoretical way. Later chapters deal with specific implementations. The objects that are represented differ in the different domains, but the representation retains its form.

What requirements can be placed on a good representation?

1. It should be easy to learn.
2. It should be useful for tasks other than learning (e.g. for recognition or manipulation).
3. It should be flexible, that is, it should be able to handle a diversity of objects.
4. It should be extensible - it should not require a great deal of effort to add new models, nor new ways of describing objects.
5. It should allow easy comparisons between models.
6. It should be compact.
7. It should allow objects to be modelled in terms most suitable to their description. It should be possible to choose the primitives and relations independently of any other objects being modelled.

Marr and Nishhara(1977) have discussed criteria for a representation of three-dimensional shape. They require the description to be inexpensively computable from the image, to be able to describe the objects within its scope canonically, and for the descriptions of objects to reflect their similarities and differences.

The choice of what kinds of low-level "primitives" to use in constructing models will usually depend on the information available, which in turn depends on the kind of vision system employed.

There are several ways of obtaining information in machine vision. The most commonly used is a single, monocular view of a scene, obtained by means of a television camera. Alternatively, one may use two offset views of the scene to obtain depth information

from binocular disparity, or by means of a range-finder, to directly produce a depth-map of a scene. Each of these input methods has its weaknesses and strengths. It is hard to obtain shape and position information from a monocular view, but monocular views have been the subject of a great deal of research and are faster to process than their rivals. Binocular vision suffers from the correspondence problem - how to match points in one view with those in the other. It does, however, offer a great deal of information when this problem is overcome. Ranging devices tend to be computationally expensive, and often do not give much more than depth information (e.g. they might not be able to see surface markings, or cracks between objects). They do, however, avoid the correspondence problem.

Using any of these devices, it is possible to retrieve edge data from a scene, or construct a set of regions (or surfaces) that make up the scene. Geometrical relationships that hold between these edges or surfaces may be harder or easier to obtain depending on the input method, but some sort of relationships are usually available. Vision has usually been concerned with analyzing some subset of these attributes.

4.2 REPRESENTATION.

A representation has been developed that is simple to construct and is powerful enough to describe a large class of objects. The representation is based on primitives and relations. Ideally, primitives could be descriptions of any scene elements that are easy to extract, have a canonical description, and have sufficient variation to allow the representation of an interesting class of objects. Relations between the scene elements may be of any suitable

nature. A small set of easily calculated relations is desired that succinctly describe the relationships between instances of the primitives in an object.

Relations are stored as schemata; their arguments are surface descriptions instead of actual surfaces. The form of the representation is a network whose nodes are primitives, and in which an arc is drawn between nodes if a relation is defined between the primitives at those nodes. Arcs are labelled with the set of relation schemata that justifies them. An object will be described by some set of these nodes and arcs, but many nodes and arcs in the network may be common to several object models. A primitive is represented by a single node, so should an instance of a primitive occur in several objects, the models produced from these objects would share that node. In addition, models may share relations, should common primitives occur in similar relationships in both models. The only thing that two models may not share is a name.

The network for the two objects of figure 4.1 is given in Figure 4.2. The objects are a cube (called "cube"), and a rectangular parallelepiped ("rect") whose model shares a side with the cube.

4.2.1 DEFINITIONS

The definitions are of the terms used to describe the representation. They are in terms of labelled graphs.

1. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of primitives. Each p_i forms one (and only one) node of the graph.
2. Let $R = \{r_1, r_2, \ldots, r_m\}$ be a set of relation schemata. Each r_i is of the form:

   <Function name, arg_1, arg_2, ..., arg_n, value>.

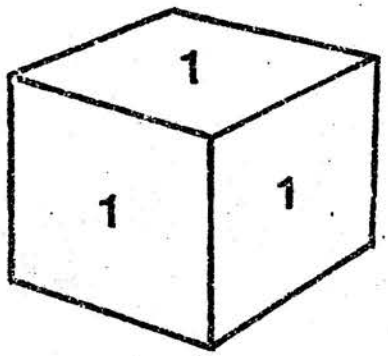
The arguments arg_1, arg_2, ..., arg_n are typed variables which may take as values instances of individual elements of P (possibly repeated). The value FALSE (0) is not considered legal for a relation schema. The labels on the arcs of the graph are sets of relation schemata, subsets of R, called label sets.

3. Let $M = \{n_1, n_2, \ldots, n_k\}$ be a set of names of models.
4. The graph obtained from modelling objects is called the graph of models.
5. A model in the system is represented by a subgraph of the graph of models. The nodes in the subgraph are those primitives that occur in the model. The arcs in the subgraph are those that have relation schemata in their label sets that occur in the model. The label set of an arc in the subgraph consists of the subset of relations that occur in the model.

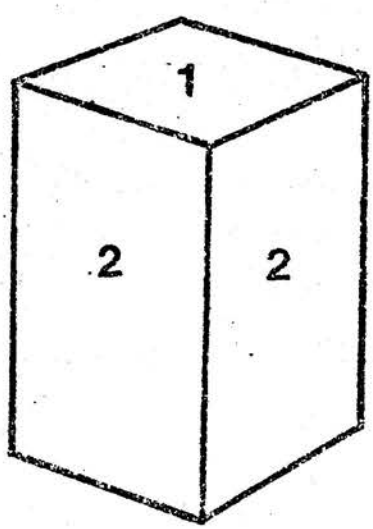
A rich indexing structure is maintained by the representation. Each primitive indexes the models in which it occurs. If a primitive forms part of the description of an object, it must index the model for that object.

Relation schemata index the models in which they occur and the primitives that form their arguments. To test whether or not a relation is satisfied requires finding instances of its arguments, while to derive any benefit from such a test requires that the information obtained be available to the models that can use it.

The models index both the primitives that form their parts and the relation schemata that describe how the parts are related.

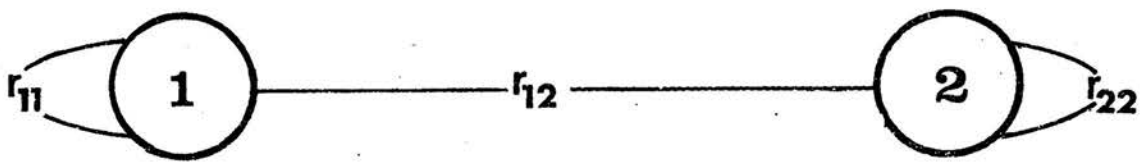


"cube"



"rect"

Figure 4.1



r_{11}= {<ADJACENT P1 P1 1>, <RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>}

r_{12}= {<ADJACENT P1 P2 1>, <RELANGLE P1 P2 90°>}

r_{22}= {<ADJACENT P2 P2 1>, <RELANGLE P2 P2 90°>, <RELANGLE P2 P2 180°>}

Figure 4.2

4.2.2 EXAMPLE.

The primitives in the example of Figure 4.1 are descriptions of the surfaces of the objects "cube" and "rect". A cube has only one type of surface (denoted 1 in the figure), while a rectangular parallelepiped has two (denoted 1 and 2). The primitives that describe these surfaces form the nodes of the graph, one node for each kind of primitive.

Two binary relations are used in the example: ADJACENT states that surfaces described by the primitives may be adjacent. That is, a single edge of the one surface is common with a single edge of the other surface. RELANGLE gives the angle of one surface from the viewpoint of the other. Note that arguments in the relations may be (perhaps different) instances of the same primitive.

The name of the model for the cube is "cube", and that for the model of the rectangular parallelepiped is "rect". The relation schemata each index the models for which they are valid. For example, while <RELANGLE P1 P1 90°> indexes only the cube model (two surfaces of type P1 can be at 90° in the cube model, but not in the rectangular block model), <RELANGLE P1 P1 180°> is valid for both the cube and the rectangular parallelepiped. Note that where surfaces were not ADJACENT, no schema was constructed. For example there is no schema <ADJACENT P1 P1 0> for the cube model. This is because relations that are FALSE (or have value 0) are discarded.

Thus, the cube has primitive set

P = {P1}

and the set of relation schemata

R={<RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>, <ADJACENT P1 P1 1>},

while the rectangular block has primitive set

P = {P1, P2}

and the set of relation schemata

R={<RELANGLE P1 P1 180°>, <RELANGLE P1 P2 90°>, <RELANGLE P2 P2 90°>, <RELANGLE P2 P2 180°>, <ADJACENT P1 P2 1>, <ADJACENT P2 P2 1>}

These sets, together with the model names, are all that are necessary to describe the objects.

4.3 ADEQUACY OF MODELS.

Having constructed a model of an object, it is useful to know how well it describes the object. A model may be inadequate in this respect because it is not sufficiently restrictive to describe only the particular object it represents. Alternatively, it may be too particular, requiring more information than is useful for the domain. A means of deciding the adequacy of a model can help to resolve these problems.

A model will be considered adequate if it is possible to construct an instance of the object represented from the information available in the model. It is clear that adequacy is to a large extent dependent on the choice of primitives and relations, and the way they are combined. A method of deciding the adequacy of a model has been developed. It is not implemented, but serves as a hand check on the utility of a model.

The algorithm performs a tree search of all instantiations of primitives and all relationships between primitives. For a model to be adequate, the algorithm must discover a closed object that is an instance of the model. A necessary assumption is that the objects being modelled encompass a closed volume of space. This assumption is used to constrain the possible positions of primitives.

The method involves collecting all relation schemata with the same function name into classes. It works by instantiating primitives successively, in an arbitrary coordinate system. The instantiations are constrained by means of the relation classes. A relation schema with arguments arg_1, arg_2, ..., arg_n might have several possible values (e.g., in the cube model, the class of relation schemata obtained from the function RELANGLE can have values 90° or 180°, with the same argument types). Instances of primitives are required to satisfy only one value in this class.

The algorithm is in four steps:

1. Collect all the relation schemata in the model. Form classes of these according to the relation name, containing all the different expected values.

   For example, in the cube model, we form the classes

   {<ADJACENT P1 P1 1>}

   {<RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>}

2. Choose one of the primitives from the model. Instantiate it in some fixed place in an arbitrary coordinate system, to represent the first surface of the object.

   Let n (=1) be the current number of primitive instantiations (ie surfaces in the object).

   Let m (given) be the total number of instantiations to be allowed for the model instance, that is, the total number of surfaces in the object.

3. We now instantiate more primitives from the object model to form further surfaces of the object in a consistent way. We proceed by choosing a relation class and insisting that the new surface be related to as many as possible of the previous surfaces in a

way allowed by at least one member of the class. This is repeated for all relation classes.

The aim is to constrain the position of the new surface as much as possible. A relation schema that has the least number of uninstantiated primitives amongst its arguments is chosen. (If possible, a schema with only one uninstantiated argument is chosen). If there is no such schema, then back up to the last instantiation. If all possible instantiations have been tried, then there is no model of size m.

If all the relations are completely instantiated and the newly-formed object is an example of the object that was modelled, then the model is adequate. Otherwise, if the new object is not a closed object, and if $n<m$, instantiate another primitive, and try again.

Those relations in the class that have the chosen set of arguments will each have an expected value. The set of these values forms the constraint on the new surface. If the constraints are unique up to symmetry, choose any one.

For example, after one surface has been assigned to the cube instance the second surface would be constrained by the relation class RELANGLE to lie at either 90° or 180° to the first.

If any new constraints are inconsistent with the old constraints, these new constraints should be ignored. This is a consequence of the fact that relations are not required to hold between all possible surfaces in a model.

4. Repeat step 3 until all the surfaces have been constrained by all the relation classes.

To find admissible models of unknown size, iterate the algorithm

with increasing values of m. Note that in this case there can be no guarantee of termination, since it is possible to construct objects out of arbitrarily many primitives.

As an example of the process, the cube model of Figure 4.3 will be shown to be adequate. For the sake of brevity, assume that it is known that a cube has six sides.

Set up arbitrary coordinates, and instantiate one of the primitives to form a surface of the cube. In this case, there is only one primitive, primitive P1. Construct a surface in a fixed place (Figure 4.4a). Call this surface A.

Now instantiate the primitive again, to form surface B. The position of this surface with respect to surface A is constrained by relations in two classes:

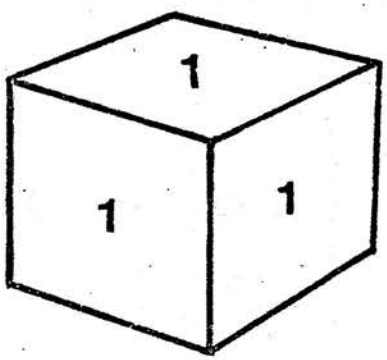
ADJ = {<ADJACENT P1 P1 1>}, and

REL = {<RELANGLE P1 P1 90^{o}>, <RELANGLE P1 P1 180^{o}>}.

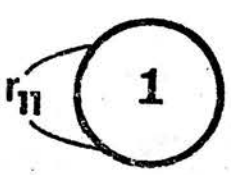
Suppose we first look at the class REL. The new surface is constrained in this case by both members of the class because both relations have the same type of arguments. B can therefore be at either 90^{o} or 180^{o} to surface A. This is insufficient information to specify surface B, but the relations in the class ADJ have not yet been taken into account.

ADJ requires A and B to be adjacent. This requirement implies that A and B must be at 90^{o}. B still cannot be positioned exactly, although it is constrained to four possible positions (one for each side of A). These are all equivalent up to symmetry, however, so one is arbitrarily chosen and surface B is fixed (Figure 4.5b).

There are no more classes to iterate over, so another

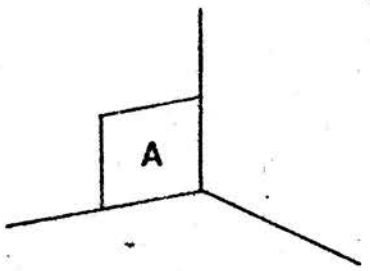


"cube"

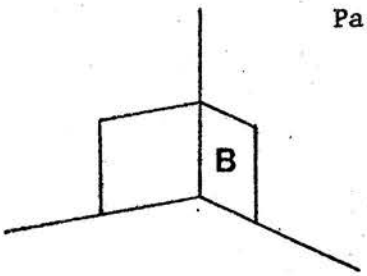


r_{11}= {<ADJACENT P1 P1 1>, <RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>}

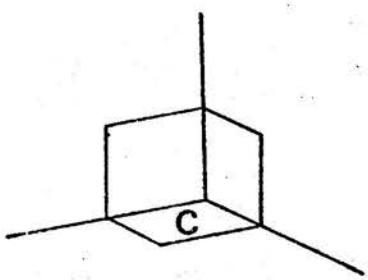
Figure 4.3



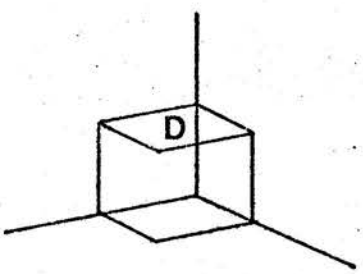
(a)



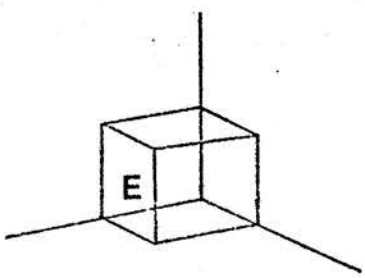
(b)



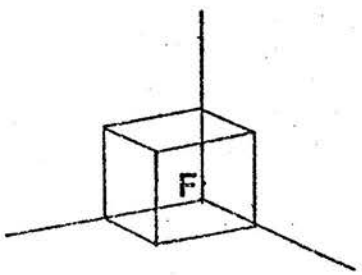
(c)



(d)



(e)



(f)

Figure 4.4

instantiation is made, called surface C.

Although instances of the schemata <ADJACENT P1 P1 1> and <RELANGLE P1 P1 90°> have been satisfied, it does not mean that these schemata need not be considered again. It is necessary to continue trying to satisfy all the schemata until a contradiction is encountered.

Similarly to B, ADJ constrains C to be adjacent to A, and REL requires C to be at 90° to A. However, C is also required to be adjacent to B, and at 90°, and this forces a specification of C complete except for symmetry. Once again, an arbitrary choice of one of the two possible positions is made (Figure 4.5c).

Surfaces D, E, and F are similar. The process will be illustrated with surface D.

As with B and C, D is required to be adjacent to A (from ADJ), and at 90° or 180° (from REL). Similarly, D is required to be adjacent to B and to C, and at 90° or 180°.

It is impossible for D to be adjacent to A and to B and to C, and one of these relation constraints will be ignored (the last one found). Once this is done, it is found that D is uniquely positioned (Figure 4.4d).

In a like manner, E and F can be fitted in to the model instance (Figures 4.4e-f). At this stage we have a closed object, which is indeed an example of a cube, demonstrating that the model is adequate.

The relation RELANGLE would not have been sufficient by itself to define an adequate model because it does not require surfaces to be next to each other, but gives only orientation information. Adjacency, however, would be sufficient because it gives both

position and orientation. The orientation is obtained by successively constraining the possible angles between surfaces until they are fixed.

It is possible to define weaker versions of adequacy. For example, a model could be called adequate if it enabled recognition of instances of the object it represented relative to the set of known models. Because of the extra information in the scene, models need not contain as much information. The scene acts as a template on which the model can be fitted. In this case, the relation RELANGLE would be found sufficient to model the cube, since the positional information would be obtained from the instance.

There is also a stronger notion of adequacy. The definition of adequacy does not rule out the possibility of constructing more than one object from a model. To ensure that this possibility is excluded the algorithm for adequacy must be exhaustively applied to the entire search space. It is also necessary to know the number of surfaces that make up a model. With this addition it is possible to use the algorithm for adequacy to show the uniqueness of a model as well as its adequacy.

The procedure given above for testing the adequacy of a model is expensive. It might require a complete search of all possible combinations of relations and all possible instantiations of primitives. It is not able to decide that a model is overspecified (ie that there are more than sufficient relations to determine an adequate model). In practice, following such a procedure by hand does indicate shortcomings in a model, and suggest what sort of steps to take in the way of choosing different primitives and relations to rectify them.

4.4 NUMBER OF ARGUMENTS FOR RELATIONS.

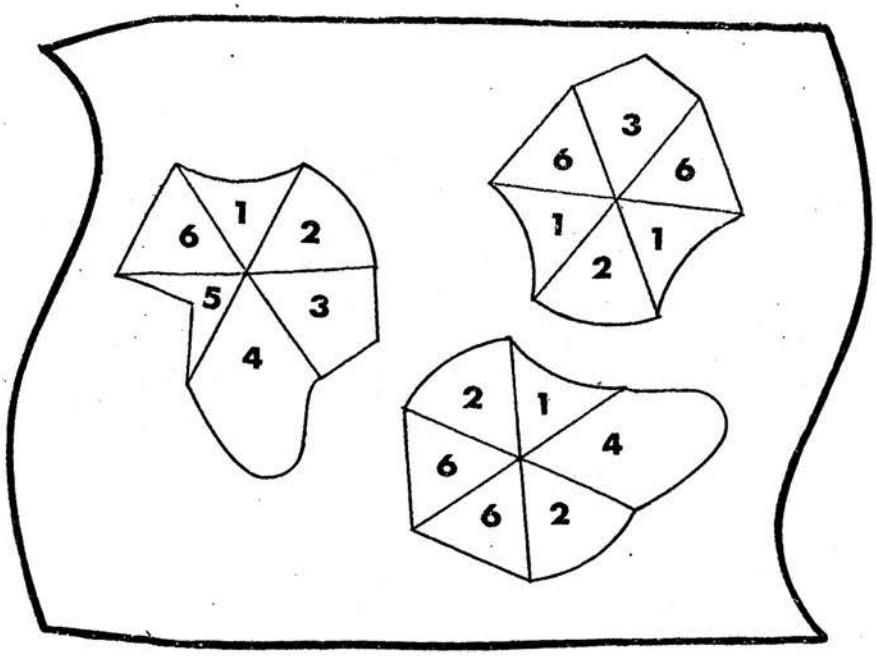
So far, relations have been mentioned in general terms, but the question of their descriptive power has not yet been addressed. It is of interest to know the theoretical limits of the representation formalism and to discover what can and cannot be represented. An interesting result concerning the limitiations of representations based on relational structure is that for any n, there exist objects that can only be distinguished using relations whose number of arguments is greater than n. This implies that no representation based on primitives and relations can be adequate for all possible models, with a finite set of relations and primitives. The set of primitives and relations used to represent objects cannot be chosen once and be expected to hold for all objects. It also means that, in general, binary relations are not sufficient to describe objects.

Consider the diagram in Figure 4.5 (I am grateful to Peter Suzman for this example). The problem is to distinguish object A from object B.

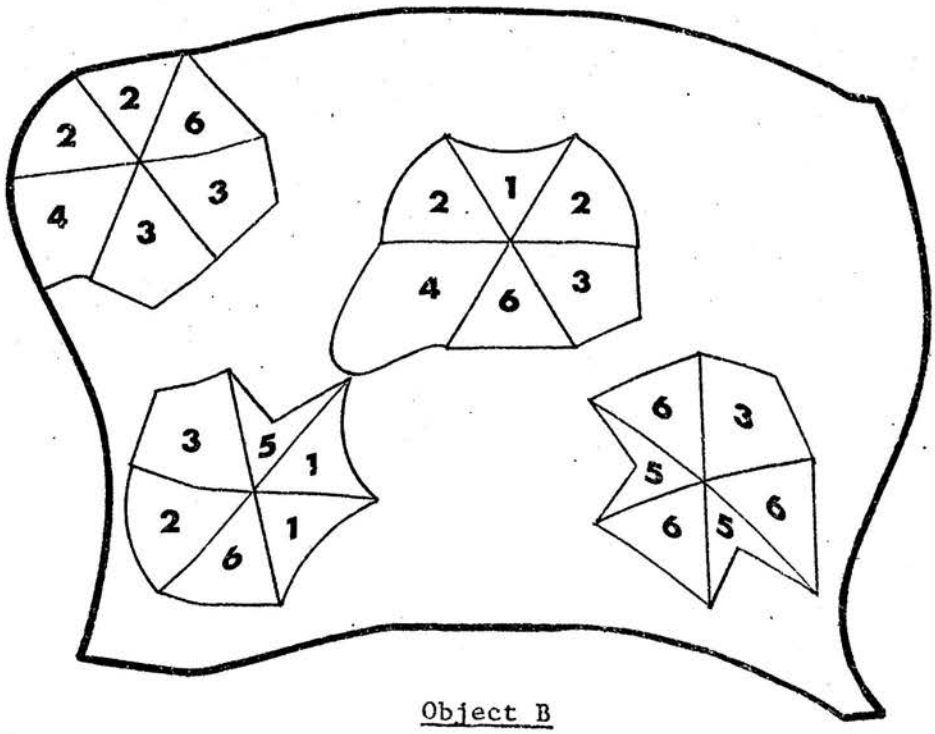
Suppose that we have modelled object A and object B, and that they have the following characteristics:
Each is made up of many facetted structures, utilizing the same descriptive primitives (e.g. primitives 1-6 in Figure 4.5).

Object A has one structure for each possible combination of the primitives. For example, there will be one structure made up entirely of facets of type 1, one with all of type 1 except one, which is of type 2, etc.

Object B differs from object A only in only one particular - it does not have a structure containing one of the combinations.



Object A



Object B

Figure 4.5

For example, it might not have the combination <1,2,3,4,5,6>.

In addition, we may assume that each structure of facets may occur many different times, so that all inter-structural relations may appear in both objects.

Now, consider what happens when we are shown a view of object A that contains the distinguished facetted structure (ie <1,2,3,4,5,6>). The only way we have of unambiguously identifying the scene is to discover this distinguished structure, which is the "signature" of object A.

Suppose we have relations that can have at most n-1 arguments (and that we do not allow conjunctions of these relations). In this case, it has been so arranged that any relationship that can be tested in the scene will appear in both objects. This is because all combinations of n-1 primitives occur in all possible relationships in both objects.

With n-ary relations, however, the structure unique to A can be identified, since that particular relationship between n primitive instances occurs only in A.

The argument is clearly true for any number of primitives, and can be demonstrated by changing the facetted structure accordingly. This proves that for every n, there exist objects that can only be distinguished using relations whose number of arguments is greater than n.

One conclusion which can be drawn from this result is perhaps non-intuitive. It follows that improving the resolution of a vision system, and thus allowing a greater discrimination between primitives, will not necessarily result in improved discrimination in recognition. This will only be guaranteed by improving the relation

set as well.

4.5 DISCUSSION.

The representation presented above depends on a different concept of "model" than is usual in vision work. Instead of describing individual parts of objects separately, those parts that have the same description are merged together in a single representation that describes all of them simultaneously. The resulting description retains most of the information about the objects, but it must be possible to separate out the individual parts later. To be able to do that requires adequate indexing, and each description must be labelled by the models in which it occurs.

Relations have to be integrated into this generalized framework. They are also generalized, forming relation schemata. The arguments of relation schemata are typed - types being generalized part descriptions. A schema thus represents a number of specific relationships. All individual parts whose descriptions are the same may be related by the same relation schema. Of course, the parts of different objects are related in different ways, and it is necessary to know which of the relation schemata are relevant for which of the models. Thus, the relation schemata must, like the generalized part descriptions, be labelled with the set of models for which they are valid. These labels act as indices into the graph of models. In Chapter 6 when recognition is discussed, it will be seen that the index structure can be very useful for object identification.

One advantage of the representation is its generality. It is possible to model any objects that can be split into parts reliably, and for which relations can be defined between the parts. Thus,

Chapter 7 presents examples of the vision system, where the representation is based on surface descriptions and relations like adjacency. Chapter 8 shows how the same representation and recognition system can be applied to spelling correction. In the latter case, the primitives are the letters in the words, and relations define the order of the letters in the words. The database of object models and of word models is a single structure. Some of the models represent objects, and some represent words. The modelling and recognition in either domain are unaffected by the presence of nodes in the structure from the other domain. Generality of this sort is useful if very different objects are to be modelled.

If the domain in which the system will work is to be changed, only the programs that produce the primitive descriptions need be rewritten or extended, and a set of functions provided that produce the relations in the new domain. This is a much easier task than changing the entire representation, which would have been required if the representation of earlier systems like that of Roberts, Falk, or Grape had been used. In fact, the basics of the spelling correction system presented in Chapter 8 were developed from the vision system in one afternoon. Later work was concerned mainly with making the input and output look more natural.

Different objects within a domain may also be modelled in different ways. Thus, in Chapter 7, the relations used to model some of the objects (e.g. the pyramid) were different from those used to model the others. This ability can be useful if an object is not naturally described in the same terms as are other objects in the domain.

The compactness and shared structure of the representation allow

efficient matching and make comparisions between models easy. Simply by indexing into the structure using some feature of a scene, all objects that share that feature are discovered. Not only that, but the graph of models indicates a set of tests to apply to discriminate between the interpretations. Instead of always applying the same set of discriminating tests, some of which would be irrelevant to the interpretations in question, a more specific set of tests, tailored to the models is made available. As a result, fewer tests may suffice to discriminate between object interpretations. A further gain in efficiency results from the fact that all interpretations are treated at once. Tests that provide evidence for more than one interpretation need only be applied once.

Looking for similarities or differences between objects involves examining the structure of the graph of models. Parts of the graph that are shared by the models of the objects indicate similarities, and parts that are not indicate differences. This is only true, however, if both objects were modelled using the same primitive descriptions and the same relations. Objects that are described differently are not directly comparable. If it is desired that comparisons be made between objects they should be modelled in the same way.

The representation does not lend itself to that paradigm of matching in which a structure is first built up from the scene, and then matched with the model structure. Such a scheme requires that the scene structure be built without reference to the model structure, and this in turn implies that the primitives and relations used to describe the parts of the scene be uniform for all the parts. If this paradigm were followed, as it has been by many authors (e.g.

Barrow et al(1972), Nevatia(1974)), the models in the graph of models would all have had to be constructed from the same primitives and relations. If we believe that different objects are best described in different ways, or if we want to use the same structure to represent different domains, this constraint becomes too limiting. The paradigm has flaws that make it a dubious method even if one has a uniform representation. By constructing the scene structure before matching one is unable to take advantage of the structure of the objects in the scene and must calculate relationships between parts of the scene that may be unrelated. It also becomes necessary to form a structure representing the whole scene, although parts of the scene may be irrelevant. A representation does not lose a powerful tool for structure matching by not allowing this paradigm.

The question of symmetry highlights another advantage of the compact representation. Earlier systems have either been unable to deal with symmetry at all (Underwood and Coates (1972)) or have handled it in a grossly inefficient manner. Each possible correspondence between parts of a symmetric object and parts of its model has been found and then one of these correspondences has been chosen as the interpretation of the object (Barrow et al (1972)). The compact representation of this thesis requires only a single structure to describe all the structurally identical interpretations. Thus the problem of symmetry disappears entirely. If a system is to work with man-made objects, it should have the ability to handle symmetry efficiently.

A constraint imposed on the representation is that it must allow the easy construction of object models from visual data. The next chapter shows how this process has been implemented. That the models

are easily constructed is partly a result of the simple components that make up the models and partly because of the structure of the graph of models. All earlier systems that have constructed models automatically have made use of a simple set of model components. Thus Barrow and Popplestone(1971) used region descriptions and relations between them to describe objects, Winston (1975) used a predefined hierarchy of object types and relations, and Underwood and Coates(1972) used surface descriptions and connectivity relations in their object descriptions. It is not obvious how more complex descriptions could be produced automatically. The main problem seems to be the lack of a uniform structure for the models of different objects.

Turner (1974), for example, represented his objects in a hierarchy. An object was described in terms of sub-objects and their relationships. Sub-objects were again broken down, until only primitive elements were needed to describe a sub-object. Turner stated that it would be hard to construct a good hierarchy automatically, because of the problems of deciding how to break objects into parts, and of choosing the kinds of parts that would be most useful.

Representations such as those used by Waltz(1975) are beyond the current state of the art of automatic modelling. His representation consisted of tables of legal junction relationships for classes of objects. The insight necessary for producing such tables would be very hard to automate. In addition, a new class of objects might affect already-existing models, for instance by requiring a new label type to be defined.

Minsky(1975) proposed frames as a way of representing

information, with the property that chunks of relevant data are available together when the correct frame is instantiated. A major problem with this approach is the effort it takes to choose the correct frame. This is tantamount to solving the recognition problem. It is also not clear how to learn new frames, and how to link them with existing frames. Problems of this nature have been largely ignored by frame systems proposed for vision (e.g. Kuipers(1975)).

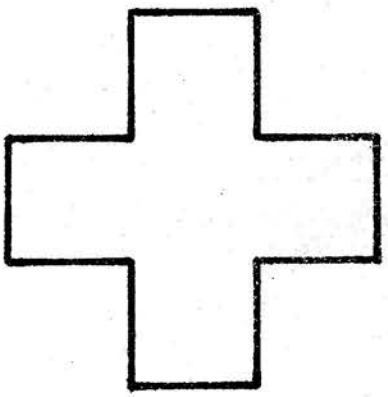
There are a number of difficulties with the representation in this thesis. One of these is shared by all representations to some extent. This is the problem of choosing good primitives and relations. Invariably this choice has been largely intuitive, with the domain of objects and the kind of input mechanism acting as the major constraints on the choice. A system employing an edge-finder naturally bases its representation on edges and their relationships, while the models for a surface-finding system will be described most naturally in terms of surface descriptions.

When relations are considered, the dimensionality of the data produced by the sensing equipment seems more important than the kind of primitive descriptive elements. Thus, if the data is one-dimensional the relations describe ordering relationships such as "precedes" or "follows". In two dimensions, adjacency and relative position become available, but the choice of what set of relationships to use is much harder. Three dimensions make position and relative orientation information available, but make the choice of a useful set of relations harder still. It is often simple to choose a set of relations to describe a particular object. In the current work, the operator makes the choice for each model from a set

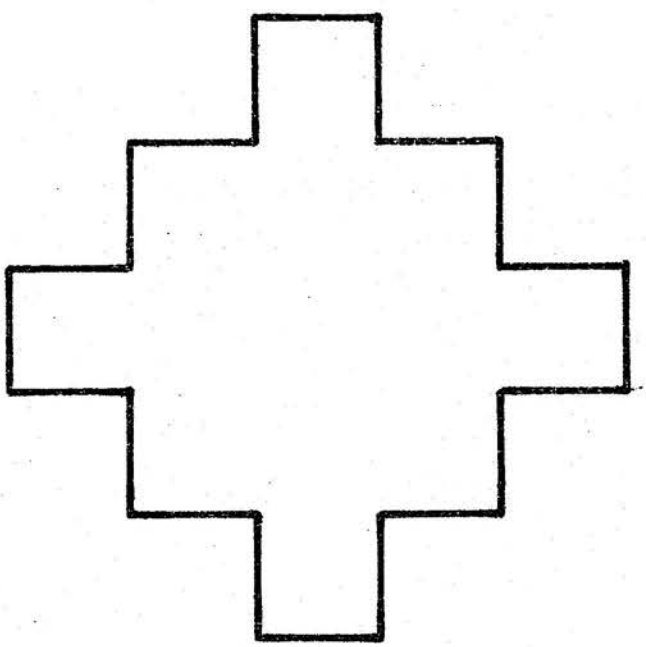
of predefined possibilities. It would be useful to be able to make this choice automatically.

A problem with choosing the relations individually for each object model is illustrated by the examples of Figure 4.6. Suppose we are using line descriptions as primitives, and wish to model the (two-dimensional) object in Figure 4.6a. We might choose simple connectivity relationships and produce a model consisting of a single line description and a single connectivity relation schema. This is clearly not an adequate model as defined in section 4.3, because it describes the object of Figure 4.6b as well as that of Figure 4.6a. Next we might try angle information, requiring the edges in the object to be at 90°, 180°, or 270° to other edges. This still does not distinguish between the objects in Figure 4.6a and Figure 4.6b. Finally, we would be forced to use relation schemata describing the distance between pairs of edges, or to use relationships between more than two edges in the object.

This problem of description would be made much less severe by adding to the description of an object the number of primitive elements that occur in it. We could add to the model for Figure 4.6a that it consists of six edges, and to the model for Figure 4.6b that it has twenty edges. This does not remove the problem, however. It still remains for any system based entirely on primitives and relations. In practice, there was no need to know the number of surfaces in each object in the domain of objects used for machine vision described in Chapter 7. Because objects are solid, it is never possible to see all their surfaces at once. The number of letters in a word would, however, have been useful information for the system described in Chapter 8 when words are being recognized.



(a)



(b)

Figure 4.6

It would require only a small amount of work to add the information to the models.

Another problem arises if models of specific objects, such as a cube, and models of classes of objects, such as the class of parallelepipeds, coexist in the graph of models. In that case, whenever the cube model is matched, so is the parallelepiped model. Unless the system knows that a cube is an example of a parallelepiped, it will not be able to identify the object uniquely (unless the cube model is more precise than the parallelepiped model). It would be useful to have relations between models as well as between parts of models. Winston was able to deal with this kind of situation because of the hierarchical structure of his models. His system was designed for one particular domain, and he was able to predetermine the hierarchy. The ability to distinguish between general and specific objects has been sacrificed in the present system to enable it to handle more than one domain and because of the requirement that all information be acquired automatically. It would be possible to create a hierarchy by making models into nodes in the structure, with relations such as "subset", but it would be hard to do this automatically except in restricted domains.

The representation could also be criticised for its lack of descriptive power. The available descriptive elements are only the primitives and relations, and all information about an object has to be stored in this form. Quite a lot of useful information does not readily fit into such a format (although this is the way most information is stored in predicate calculus). For example, it is often useful to know the volume of space occupied by an object (e.g. for trajectory calculation). This information concerns the object as

a whole, rather than its parts.

The representation was conceived as only part of the total object description - the part that would enable easy modelling and recognition. It was expected that there would be another class of information available about the object that could be accessed when it was recognized. This is a reasonable assumption because a lot of necessary information about an object cannot be extracted merely by looking at it, for example, its weight.

When an assembly task is being programmed, the form of the models of objects will be determined by the kind of equipment used in the assembly and the task at hand. Assembly is concerned with bringing parts of objects into specified relationships (Popplestone (1977)). The parts need different descriptions for recognition and for manipulation. For example, it will often be useful for a manipulator to know that two surfaces of an object occur on opposite sides and are parallel. This knowledge would enable the manipulator to pick up the object by grasping those surfaces. The relationship would not, however, be useful in a vision system because both surfaces could not be visible simultaneously. The description of objects for assembly purposes (e.g. using the RAPT language (Popplestone et al (1978)) is best done when the assembly task is specified. A system that can acquire models for recognition, and can use the recognition system to access the data that are more useful for assembly, is an important component of a complete system.

Most of the disadvantages of the representation stem from its ability to work in more than one domain of objects, and its ability to acquire object models automatically. The advantages outweigh the disadvantages they engender, specially since the disadvantages are

common to many earlier systems.

In summary, a representation for modelling objects has been presented and discussed. The representation has the novel features that models are compact and are distributed in a single network. Some of the advantages to be gained from the representation have been dealt with. More of them will become apparent when recognition is discussed in Chapter 6. The next chapter presents an implementation of the representation for the domain of machine vision.

5.0 BUILDING MODELS IN THE VISUAL DOMAIN

The previous chapter presented a representation in a theoretical form. This chapter presents the version that has been implemented. It shows how models are constructed using a number of views of the object, and the kind of information that is available after the models have been learned. To enable a fuller understanding of the domain and the kind of information to be used, the ranging system will briefly be described.

5.1 THE RANGER

The ranger consists of a ranging device and several computer programs for interpreting its output (Popplestone and Ambler (1977)). The ranger hardware has a projector that casts a plane of light onto a scene on a table. When viewed by a TV camera from a point at an angle to the plane, points on the plane can be located in space by triangulation. The table is able to translate in a horizontal plane, and supports a turntable that can rotate. The configuration is depicted in Figure 5.1 (from Popplestone et al. (1975)).

A stripe of light cast on a scene will appear to the TV camera as a broken curve, or a set of straight line segments, depending on the type of surface encountered. The image is such that points

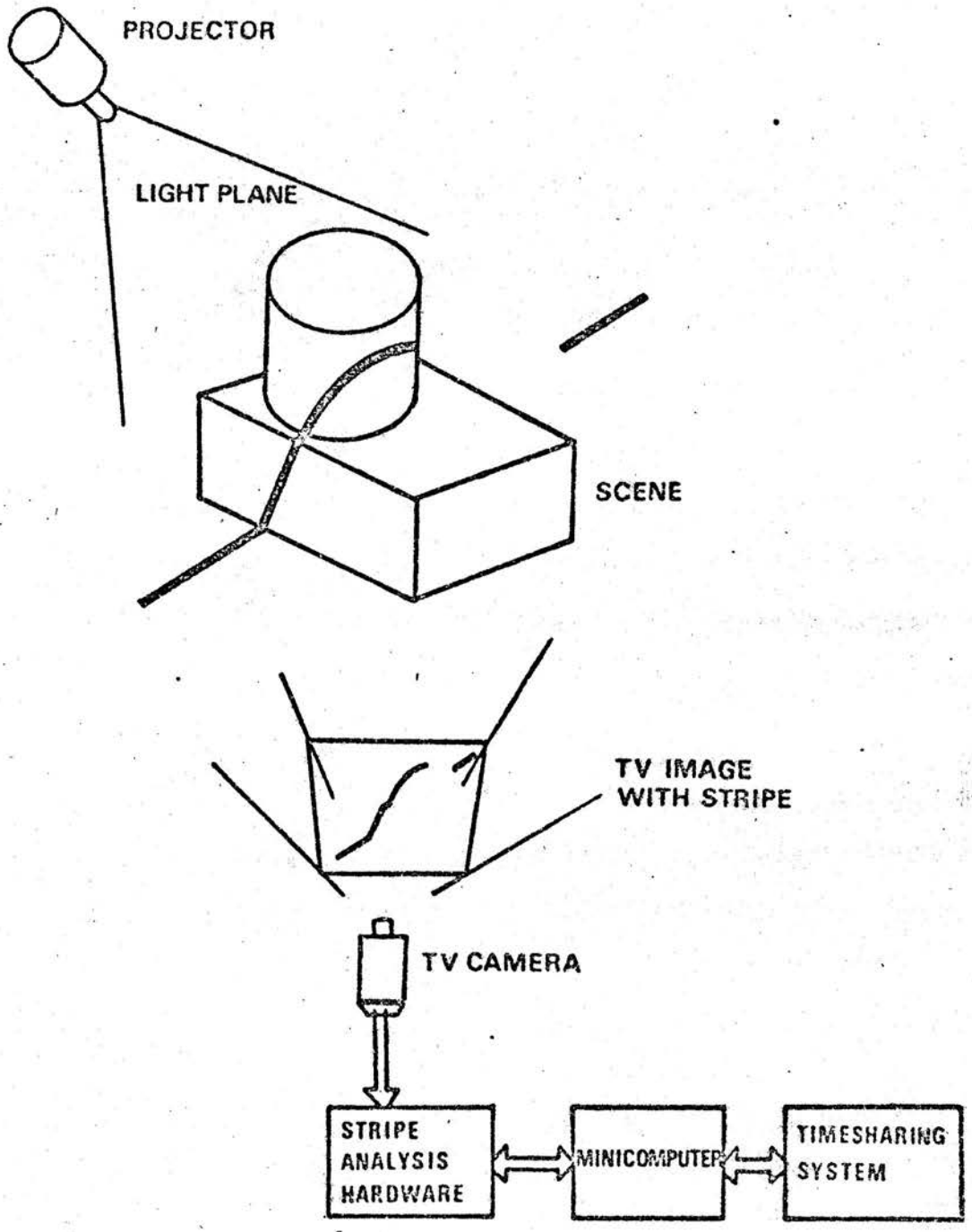

PROJECTOR
LIGHT PLANE
SCENE
TV IMAGE
WITH STRIPE
TV CAMERA
STRIPE
ANALYSIS
HARDWARE
MINICOMPUTER
TIMESHARING
SYSTEM

Figure 5.1

appearing to the right of the TV screen are further away from the camera than are points to the left.

A stripe gives only a single cross-section of a scene, so a set of cross-sections is taken at fixed intervals across the scene. Such a scan provides a range map (see, for example, Figure 3.2). The next stage is to dissect the map into surface elements obeying some geometric law, in this case into planes and cylinders. This is accomplished in two stages, a control program, resident in the DECsystem-10, interacting with the segmenting program in the minicomputer (a Honeywell H316) that controls the ranging device.

The segmenter first reduces the amount of information stored by expressing each stripe as a series of straight line segments, some of which will be labelled as smoothly joined (if there is no significant change of curvature at their junction). Segments on the curved portions of consecutive stripes are aligned so that the plane finder will divide them into planar facets for further analysis. The data produced by the segmenter are sent to the DECsystem-10, one stripe at a time, as they are processed by the segmenter. No more than two stripes will be present in the minicomputer at one time, so that curved sections are faceted on a local basis. The resultant problem of broken facets around an irregularity is left to the control stage to sort out.

When a complete scan has been sent to the main computer, the process of finding the surfaces can be carried out. If a whole object is to be analyzed, several scans must be taken, with the object rotated on the turntable between each scan. The surface-finder handles multiple scans one at a time, and merges the results.

Plane faces appear as clusters of parallel, equidistant line segments in 2-D picture space. They are discovered by a histogram method based on the direction of the segments, and the distance between them. Clusters are unified into single faces if they have the same (planar) equation.

Having found the planar faces, those marked as smoothly joined must be re-examined and an attempt made to describe them in cylindrical terms. A function minimization technique is used, which finds a best cylinder axis and radius to fit the data.

The final output of the ranging system is thus a set of planar or cylindrical surfaces, oriented and positioned in three dimensions.

5.2 THE MODELLING PROCESS.

5.2.1 EXTRACTION OF SURFACES.

An object to be modelled is placed on the turntable. The object is passed under a plane of light, and scanned at fixed intervals. When the scan is complete, the object is rotated by a known amount, and scanned again. This process is repeated until all desired views of the object have been scanned.

The result of processing the range map obtained in this way is a set of surfaces, each described by:

1. The surface type (plane or cylinder).
2. The equation of the surface.
3. The three-dimensional co-ordinates of the centroid of the surface.
4. The radius of a cylindrical surface.
5. The segments of lines that make up the surface.

Some further processing is done on this information before a

model is produced. The aim of this processing is to derive simple, non coordinate-specific information about the surface, which can be used to form primitives. The information chosen is a measure of the dimensions of the surface.

Many different ways of measuring the dimensions of the surfaces were examined. The method described here has the benefit of simplicity, and has proved robust in practice. The descriptions produced are not adequate to desribe the surfaces themselves. This situation is discussed further in section 5.3. The result of applying the method is a very crude estimate of the shape of a surface, described by only two numbers, or dimensions. The dimensions obtained are the width and the length of the surface for planar surfaces. Cylindrical surfaces are described by their length and their radius, the radius being known already.

The width of a planar surface is obtained by examining the line segments produced by the light plane that impinged on the surface when it was scanned (e.g. segments ab, cd, ef, ... in Figure 5.2a). The width could have been taken as the mean length of the segments. This measure, however, is usually incorrect because the segments are not perpendicular to the line along which the length of the surface is measured, but inclined at some angle ϕ to it. The length of the surface is measured along a line derived from the endpoints of the segments. The line is in the direction defined by most of the endpoints, leaving out any anomalous points. In Figure 5.2a, for example, points a,c,e,g,f,and h would be included, but b and d would be excluded. The endpoints are grouped into two classes for this analysis, one for those at each end of the surface. The angle ϕ is measured between this line and the direction of the line segments in

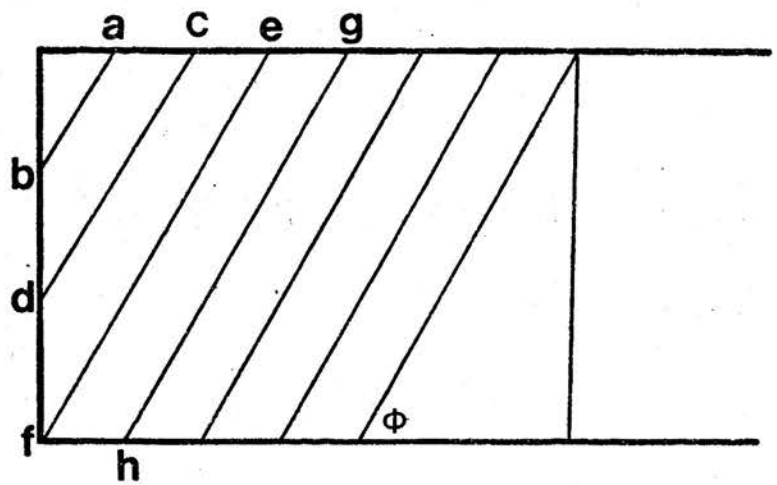


Figure 5.2a

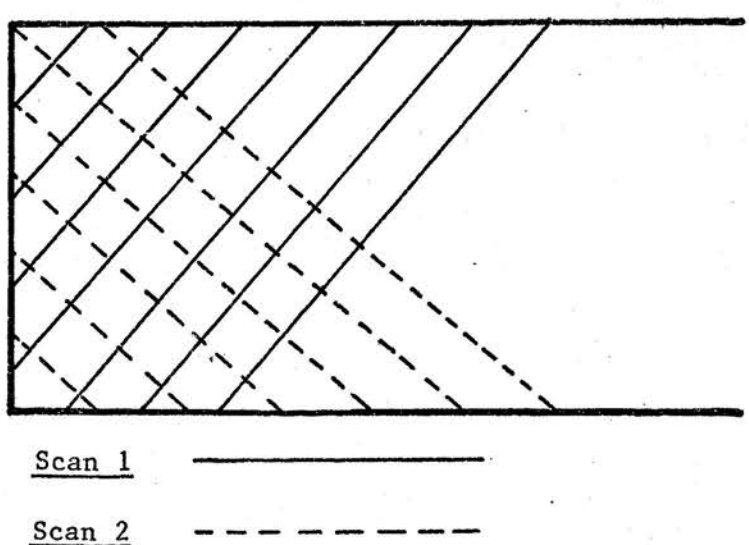


Figure 5.2b

the scan. To compensate for this angle, the mean length of the line segments is multiplied by the sine of the angle the segments make with the direction in which the length of the surface is measured.

ie Width(surface) = Mean(length(ab), length(cd),...) * sinϕ

The length of both planar and cylindrical surfaces is simpler to calculate. It involves simply adding up the distance between the line segments on the surface. In fact, the distance between line segments is usually fixed, so that the length can be calculated as the product of the number of segments on the surface and the mean distance between line segments. The distance used is not the perpendicular distance between segments, but the distance between endpoints of successive segments, ac, ce, eg, etc.

ie Length(surface) = no segments on surface * mean distance between segments.

The process of measuring surfaces is complicated by having to take several views of the surface into account. A surface may appear in more than one of the scans necessary to view the whole object. All the information is used to form the final estimate of the size of the object.

In earlier versions of the procedure, all line segments belonging to a surface were treated at the same time. It was found in practice that whereas errors in the ranging process were fairly small within a single scan, the cross-scan errors could be large. For example, the positions given for the endpoints of line segments in one scan could differ markedly from those in another similar scan. This difference is due to errors in table movement in the scanning

process. Because of this problem, each scan of a surface is treated individually and the results are averaged over all scans.

It is possible to discover when a new scan begins by measuring the angle between successive line segments. Within a scan, the line segments all have a similar orientation. When a surface is scanned from two different viewpoints, the line segments produced will have different orientations. Therefore, a change in the angle between line segments is used to signal a new scan (Figure 5.2b). Within each scan, the surface dimensions are measured as above.

From the surfaces the modelling program will construct surface descriptions, and from the relationships that hold between the surfaces it will construct the relation schemata.

The user must supply a name for the object being modelled. This name may be the same as that of a previously-existing model, in which case the existing model will be updated should any discrepancies become apparent. The user must also supply a list of the names of the functions to be used to form the relation schemata in modelling the object.

5.2.2 PRIMITIVES AND RELATIONS

From the set S of surfaces, a list P′ of surface descriptions is constructed, one for each surface. Elements of P′ may be repeated. The elements of P′ will be used to construct the set P of nodes in the graph of models. The descriptions are constructed from the surfaces by extracting the type of surface and its dimensions. All coordinate-specific information is discarded when forming the descriptions. This information is, however, still available to the rest of the system, and will be used to construct the relation

schemata. At the same time, the set

$$SP = \{\langle s,p\rangle \mid s \in S,\ p \in P'\}$$

of pairs of surfaces and their associated descriptions is constructed. SP links the surfaces in the image with the descriptions in the model world. The three-dimensional positions of the surfaces and their relationships with other surfaces will be used in building the model. The differentiation between the image and the model is crucial, and should be born in mind in all aspects of the system.

The next step is to determine the relationships between the surfaces in the object being modelled. In the implementation, relations have been restricted to having at most two arguments, for ease of acquisition and because the number of relation schemata produced for n-ary relations can be very large. In practice, binary relations are adequate for the domains under consideration.

The choice of the relations with which to model an object is left to the user, who lists the names of the functions to be applied at the time the model is constructed. The use of a relation requires that an executable function exists which can calculate values for instances of its arguments. A number of different functions have been used in experiments. Some of these are:

ADJACENT: If the light plane in some subscan passed over both surfaces at the same time, and the line segments produced by the light plane were touching, then the surfaces are adjacent.

RELDIST: The distance from the centroid of one surface to that of another.

RELANGLE: The angle between two surfaces, from the viewpoint of the first.

Whereas ADJACENT is a true relation, in that it has values "true" or "false", RELDIST and RELANGLE can have a large number of values, each of which will give rise to a new schema when it is encountered in an object being modelled. (There will not, however, be an infinite number of such values, since a range of values which are sufficiently similar will be grouped into an equivalence class).

The schemata are constructed as follows. For each pair of surfaces in S, the chosen set of functions is applied. The result of the application is used as the expected value slot in the schema. The name of the function that was applied is the value of the function name slot, while the argument slots are filled by the descriptions that correspond to the surfaces used in the evaluation. The correspondence is obtained from the set SP of pairs of surfaces and their descriptions. Using the descriptions, rather than the surfaces themselves, generalizes the relations to make them valid for all surfaces that match the descriptions. If the value of the function is FALSE (0), a schema is not constructed.

5.2.3 PRODUCING THE GRAPH OF MODELS.

At this stage there is

1. A list P′ of surface descriptions.
2. A set of relation schemata.
3. A model name.

This information must be integrated into the graph of models. Any new information should be retained, but duplication or spurious arcs should not be introduced.

Each of the descriptions in P′ is matched against the primitives in P, the set of nodes of the graph of models. Only if there is no

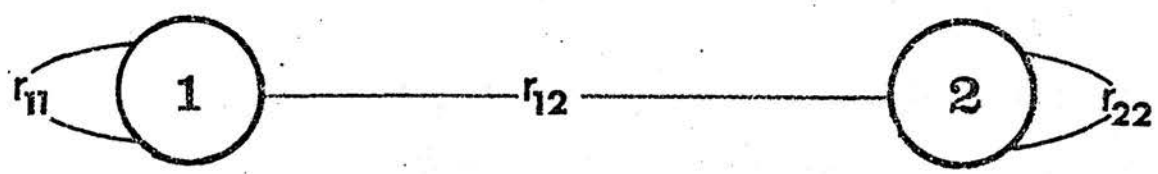
match (i.e. if a description is sufficiently different) is a new primitive node created for the description and added to P. Primitives are described in exactly the same way as surface descriptions, but there is only one primitive with a particular description. The result is a set $P \cup P'$ of primitives. Matching requires that the surface types be the same in the primitives and the surface descriptions, and that the dimensions be within some fixed tolerance of each other (currently, dimensions are allowed to vary by about 20% before the match fails).

When a surface description matches with a primitive already in the graph, the values of the primitive in the graph are updated. The mean of each dimension in the surface and the primitive becomes the dimension for the primitive in the graph.

The name of the model is added to the set of names if it is not already a member. The new set of relation schemata is added to the set R of schemata that already exist as follows. If a new schema is not already a member of R, it is added to R, and indexes only the new model. If a relation schema already exists in R, it is not duplicated. Instead, the schema is altered so that it indexes the new model in addition to any others it might already index. The model indexes the primitives and relation schemata that occur in it, and the primitives index the new model.

5.2.4 EXAMPLE.

As an example, the graph of Figure 5.3 will be constructed. The construction is in two stages, one for each object to be modelled. The result of applying the modelling program to the data is seen in Figure 5.4. The data used in this example were hand generated for



r_{11}= {<RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>}

r_{12}= {<RELANGLE P1 P2 90°>}

r_{22}= {<RELANGLE P2 P2 90°>, <RELANGLE P2 P2 180°>}

Figure 5.3

the purposes of explanation.

First consider the cube (Figure 5.5a): The following are produced when the cube is scanned. The set of surfaces

S = {A, B, C, D, E, F}, the list of surface descriptions

P′ = [P1 P1 P1 P1 P1 P1] (because the object is a cube, the descriptions are all the same - square planes), and the set of pairs

SP = {P1/A, P1/B, P1/C, P1/D, P1/E, P1/F}

The set P of nodes is obtained from P′ by finding those descriptions that do not match with primitives that are already in P - in this case P is initially empty. Because all the surface descriptions are the same, only one primitive need be added to P, so that P = {P1}. The primitive in P has added to it a pointer to the model "cube" of which it is a part.

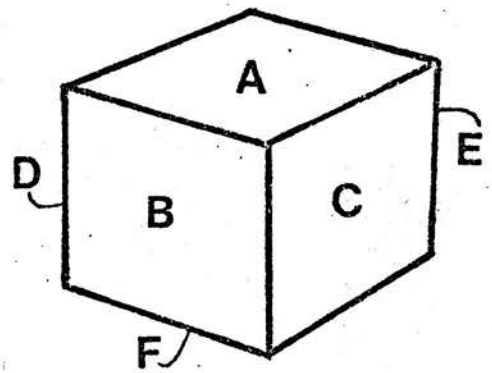
The relation schemata are obtained by applying prespecified functions to pairs of surfaces. The expected value slot is filled with the result of applying the function, while the argument slots are filled with the primitive matching the surfaces (The primitive is found from the set SP). Thus, in the cube, sides A and B are at 90°, so when the function RELANGLE is applied, the result is <RELANGLE P1 P1 90°>. This schema is added to R if it is not already a member. The set of models indexed by the schema is also modified to include the new model. The results of applying the function to all pairs of surfaces are the sets:

P = {P1}

R = {<RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>}

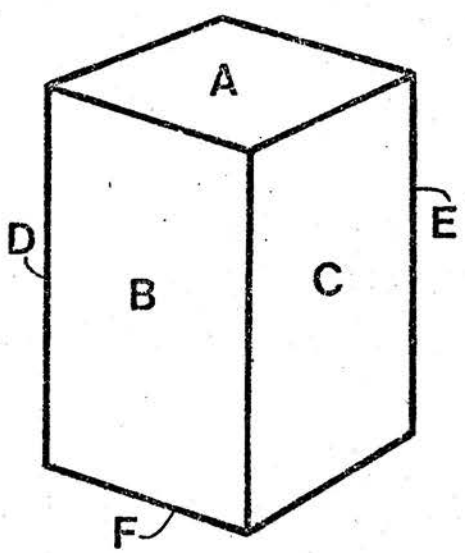
```
MODEL("CUBE",[CUBE],[RELANGLE]);
MODEL("RECT",[RECT],[RELANGLE]);

PRIMITIVES:

PLANE  P1
BODYLIST RECT CUBE
EXTENT  5.0     5.0

PLANE  P2
BODYLIST RECT
EXTENT  8.0     5.0


OBJECTS:

RECT
DESCRIPTION
PLANE  P1
APPEARS IN RELATIONS: RELANGLE R2 RELANGLE R5
PLANE  P2
APPEARS IN RELATIONS: RELANGLE R5 RELANGLE R4 RELANGLE R3

CUBE
DESCRIPTION
PLANE  P1
APPEARS IN RELATIONS: RELANGLE R2 RELANGLE R1


RELATION SCHEMATA:

RELANGLE  R2  ARGUMENTS  P1   P1
EXPECTED VALUE  180.0
BODYLIST RECT CUBE

RELANGLE  R4  ARGUMENTS  P2   P2
EXPECTED VALUE  180.0
BODYLIST RECT

RELANGLE  R5  ARGUMENTS  P2   P1
EXPECTED VALUE  90.0
BODYLIST RECT

RELANGLE  R1  ARGUMENTS  P1   P1
EXPECTED VALUE  90.0
BODYLIST CUBE

RELANGLE  R3  ARGUMENTS  P2   P2
EXPECTED VALUE  90.0
BODYLIST RECT
```

Figure 5.4



"cube"

(a)



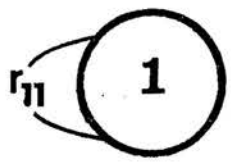
"rect"

(b)

Figure 5.5

M = {cube}

and the graph



r_{11} = {<RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>}

Now consider the parallelepiped "rect" (figure 5.5b). Scanning this from all sides produces the set of surfaces

S = {A, B, C, D, E, F},

the list P′ = [P2 P2 P1 P2 P2 P1], and the set

SP = {P2/A, P2/B, P1/C, P2/D, P2/E, P1/F}.

Because the cube has already been modelled, the graph is already partially constructed. It must be extended only where necessary.

There is already a node of type P1, so a node of type P2 is the only node to be constructed. P = {P1}∪{P1, P2} = {P1, P2}. The elements of P are updated to index "rect".

There is one relation schema common to the two models, <RELANGLE P1 P1 180°>. (Two square surfaces can be parallel in both the cube and the rectangular parallelepiped.) The set of models indexed by this schema has the model for "rect" added to it. All other relation schemata are new, and are added to R. They will index only the model for "rect".

A label on an arc of the graph is the union of all relation schemata whose arguments are the primitives joined by the arc. Thus, in Figure 5.3, r_{11} ={<RELANGLE P1 P1 90°>, <RELANGLE P1 P1 180°>},

etc.

This concludes the formation of the graph of models for "cube" and "rect".

5.3 DISCUSSION

The implementation of the modelling system follows the representation presented in Chapter 2 quite closely. The main difference concerns the restriction of the system to binary relations. This was necessary for several reasons. Firstly, it is easy to construct the schemata for binary relations because two surfaces are easier to process than three or more. Secondly, there are fewer binary relations than ternary or n-ary ones. It would be useful to have some kind of discrimination technique that could decide which relations were important for modelling an object. Instead of applying binary relations between each pair of surfaces, it would then be possible to pick out surfaces and n-ary relations to apply to them in such a way as to produce a minimal model of the object. Unfortunately, no such discrimination technique could be devised, and binary relations only were used. Despite the result given in section 4.4, binary relations proved adequate for the examples in the domains of Chapters 7 and 8.

Another problem that has not been satisfactorily solved concerns the description of surfaces. The crude dimensions used in this chapter are inadequate for describing the shape of the surfaces. There are only two numbers, corresponding to a length and a breadth, used to describe a surface. Thus, surfaces of very different shape, such as a rectangle and a triangle, may be indistinguishable. The description was chosen for its simplicity and because it does not

degrade too seriously even if surfaces are partly occluded. The relationships between surfaces often serve to discriminate between objects with similar primitive descriptions of different surfaces. The way in which a triangular surface relates to other surfaces will usually be different from the way a rectangular surface does.

The surface descriptions will be used to index the models during recognition. The descriptions used in the implementation are suited to this primary role because they are easy to construct and do not require a great deal of work to be done without the benefit of the context that the models provide. Crude descriptions index a larger number of models than would more specific descriptions. Having indexed the models, it is possible to use information associated with the models to refine the surface descriptions. Adler (1976) used this technique in his system to analyze PEANUTS cartoons. In the present system, relations and predicates serve a similar purpose.

A number of alternative surface descriptions were implemented, but were rejected for various reasons. An early version of the system used the surface outline as part of its description, in a way similar to that of Ambler et al.(1975) in the Versatile Assembly System. This representation proved too fragile when confronted with occlusion and it became very difficult to compare surfaces with model descriptions even in quite simple scenes. The problem was made even worse because of the discrete nature of the range data. The representation of Underwood and Coates(1972) would also have been unsuitable because of the requirement that complete surfaces be visible for identification. Perhaps general attributes of the kind used by Barrow and Popplestone(1971) would allow a more discriminating surface description. For the domain in which the

system was used, however, the crude descriptions sufficed.

Another alternative to surface description based on range data was used by Agin and Binford (1973), who described objects in terms of generalized cones. Their work was extended by Nevatia and Binford(1973, 1977), who were able to recognize a small set of objects viewed by a ranger. Generalized cylinders consist of an axis, which is a space curve, and a cross-section, usually of fixed shape. The problem with using such descriptions for man-made objects is highlighted by the description of a cube. This consists of a straight axis with a square cross-section function. This description does not reflect the symmetry of the cube, which is its major feature, because the description along the axis is different to that perpendicular to the axis.

There are a number of problems that arise from using the ranger as an input device. It is only possible to see those slices of an object that are illuminated, so that the degree of certainty about the shape of an object is limited by the distance between the stripes. Surface markings and cracks are not visible because the data consists only of line segments whose appearance depends on the shape of the surface.

Another problem arises because of the geometry of the viewing equipment. Certain parts of the scene are not visible from the TV camera because they are hidden by surfaces in front of them. Equally, the light plane will not fall on certain parts of the scene because they are hidden by other surfaces. Thus there is more shadowing in a range map than in a TV image produced using normal lighting. The amount of shadowing depends on the distance between the light source and the camera.

The time factor can also be troublesome. Instead of needing a single image, the ranger requires one for each stripe. Between acquiring each stripe a pause is necessary to enable the table to move and settle down. As a result, the time taken to view a scene may be about one or two minutes. Some processing can, however, be carried out while the table is moving.

When the examples of Chapter 7 were being produced, it was discovered that the surface-finding routines that process the scans were inadequate. It was necessary to improve this part of the system to enable it to separate the scan more reliably into surfaces, and to position the surfaces more accurately. A substantial part of the surface-finder was rewritten, and significant improvements were made both in the quality of the output and the speed of processing. It became possible to cope successfully with a variety of objects composed of planar and cylindrical surfaces.

In the domain of spelling correction (Chapter 8), the problems of describing surfaces and of using real visual input did not arise. The models produced were not subject to errors, so that the evaluation of the representation and recognition systems was made easier.

In summary, the implementation of the modelling system using the representation of Chapter 4 has been discussed. It has been shown how models can be constructed automatically and how generalized models are produced.

6.0 RECOGNITION

6.1 INTRODUCTION

This chapter presents and discusses the recognition system. It shows how the representation developed in the previous chapters allows a number of advantages to be gained over previous recognition algorithms. The advantages are the result of a more selective discrimination between interpretations for parts of scenes, and of a more efficient indexing mechanism.

The essence of recognition is the matching of a description of an image with a stored structure. This problem can be looked at as one of finding maximal cliques in some graph derived from the description (Barrow and Burstall(1974)), which is known to be a polynomially complete problem. The matching is harder even than the subgraph isomorphism problem because in general one is dealing with sets with properties, and more than one binary relation. Also there is usually some tolerance on matching. Recognition requires some maximal match or correspondence between the image structure and the structure of models, which preserves the properties and relations of each. An inefficient procedure can result in very poor performance due to the large search space involved. Even small improvements can prove significant in practice.

There are a number of criteria with which a recognition

algorithm should conform. Firstly, the recognizer must use the descriptions produced by the modelling system. When models are constructed automatically, a more powerful recognition system is required than might be necessary if the models were custom-built. Automatically learned models have only approximate descriptions, which are subject to error. This means that a robust and forgiving recognition algorithm is needed. In our system, individual models are allowed to be constructed out of entirely different relations and primitives. Therefore, the recognition must be flexible enough to handle all possible models. (Objects whose models were constructed using incompatible input mechanisms are not allowed to appear in the same scene).

There are other criteria for a recognition system. It should be efficient - i.e. it should do as little unnecessary work as possible, and should not repeat work. It should handle symmetry in a reasonable way. Many systems, when matching, for example, a view of a cube against a cube model, will find all possible correspondences between surfaces in the view and representations of surfaces in the model. Any one of these would be sufficient.

An important requirement for a recognition system is "graceful degradation" (Marr(1975)). The system should at all times be able to return a partial result, as specific as possible considering the evidence it has so far, but making no unjustified hypotheses.

6.2 RECOGNITION.

The graph constructed from the surfaces in a picture is called the scene graph, and the graph obtained from modelling objects is called the graph of models. A model graph is a subgraph of the graph

of models. It consists of those nodes whose arcs have relation schemata that index the model, together with those arcs and schemata.

Recognizing an object corresponds to finding a mapping from a subgraph of the scene graph to a non-trivial subgraph of the model graph. It is not practical to construct a scene graph explicitly and then attempt to match subgraphs into the graph of models. Such a procedure would involve calculating all possible relations between all surfaces in the scene, and matching a very large number of subgraphs against the graph of models. Moreover, every model would have to be defined in terms of the same relations.

Instead of creating the scene graph and then matching, the graph is created dynamically, and partial matches are used to direct the construction of arcs.

Because the data are imperfect, a richer structure of interconnections is maintained in the scene graph than in the graph of models. An element of uncertainty is included to allow tentative hypotheses to be made on the basis of knowledge local to a subgraph of the scene graph. A surface with two interpretations will have two nodes in the scene graph, linked together to show they are ambiguous interpretations of the same surface. Later, a constraint analysis procedure attempts to resolve local inconsistencies. A globally best interpretation of the scene is sought, in terms of the objects in the scene and the surfaces that make up these objects. By a globally best interpretation we mean a consistent interpretation for each part of the scene that gives the greatest justification to the interpretation of the whole scene.

During the process of recognition a scene graph is constructed, with (possibly) more than sufficient nodes and arcs. The graph is

then examined, and nodes and arcs are deleted if their presence cannot be sufficiently justified. The structure of the scene is made apparent by constructing a graph whose nodes are surfaces, rather than descriptions of surfaces. The arcs are labelled with relation schemata whose arguments have been instantiated. The graph of models is used to guide the construction of the scene graph, and ensures that the construction is efficient.

To be more accurate, the scene graph and an intermediate structure derived from the graph of models are constructed simultaneously and in the same place. All nodes in the structure are pairs, each consisting of a surface from the scene, for the scene graph, and a primitive from the graph of models, for the intermediate structure. The intermediate structure is derived from the graph of models by instantiating primitives, perhaps several times, to form a conventional relational structure representation of the objects. Relation schemata are also instantiated. The resulting relational structure is isomorphic to the scene graph. Thus the process of building the scene graph and matching it with the models is unified. The joint structure is referred to as the scene graph, although this term is not strictly accurate.

6.2.1 EXAMPLE.

Suppose we have the graph of models built earlier (Figure 5.3), and want to recognize the scene in Figure 6.1.

We cannot be sure of always matching picture surfaces uniquely against model descriptions, so a picture surface is allowed to match with several primitives in the graph of models and a matching confidence is associated with each correspondence. For convenience

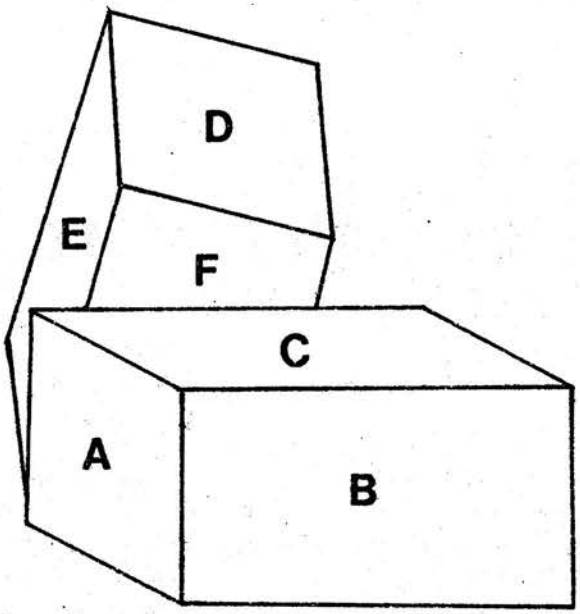


Figure 6.1

in this example, the matching confidences will be ignored. When a picture surface has matched with several primitives, these primitives will be called linked, and an arc will connect the nodes corresponding to them in the scene graph. Underlining in the set SP below, and dotted lines in the graphs, will indicate linked primitives.

When the example scene is scanned, the following information is obtained:

S={A, B, C, D, E, F}

P'=[P1 P2 P2 P1 P2 P3]. Surface description P3 has no analogue in the graph of models, because surface F is occluded. The occlusion is noted (see below) and description P3 is matched with all primitives whose dimensions are larger than those of surface F. This gives the set

SP={P1/A, P2/B, P2/C, P1/D, P2/E, P1/F, P2/F}

Surface occultation is detected in two ways. During the processing of surfaces to find their dimensions, a check is made for a sudden change in the length of line segments. Such a change indicates occlusion if the light plane passed over another surface in the same subscan and the line segments produced in the subscan are touching. Alternatively, if a surface cannot be matched with any primitive, it is assumed that the surface is incomplete. In both cases the surface is matched with all primitives larger than itself, with diminished confidence.

We want to associate primitive/surface pairs in SP with models in the graph of models in such a way that we have greatest global confidence in the associations. It will not always be possible to associate pairs with a single model instance, so that ambiguous

results are possible.

Each association causes assignment of a node in the scene graph. Nodes have an interpretation as part of a distinguished subgraph of the scene graph associated with a particular model. They also have reasons justifying their interpretations. The reasons are relations that denote compatibility between a node and other nodes in the same subgraph. The relations label arcs that join nodes in the subgraph. When all possible assignments have been made, any ambiguous interpretations will be adjudicated on the basis of the reasons for assignment. This gives rise to a kind of filtering procedure.

First, however, we must make the assignments. We want to find what model names to associate with each member of SP, in cases where a primitive indexes more than one model.

As soon as at least one member of SP is associated with a model, a distinguished subgraph of the scene graph will be set up for an instance of that model. Assignments to a subgraph are made in two ways:

1. There is no evidence to contradict the assignment. That is, the node is the first to be assigned in the subgraph for the model.
2. There are nodes p_i/s_i, p_j/s_j, ... already assigned to a subgraph of the scene graph. If the primitive p_k of the pair p_k/s_k to be assigned to the scene graph has arcs in the graph of models connected to some of p_i, p_j, ... then there is the possibility of assigning p_k/s_k to the same subgraph of the scene graph. The pair p_k/s_k will be assigned to it if some relation schema on the arc connecting p_k to one of p_i, p_j, ... is satisfied, and that schema indexes the model associated with the subgraph.

   Instantiated relation schemata form the reasons for an

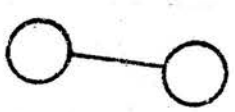
assignment. They form the labels on the arcs of the scene graph.

Suppose we choose first to assign P2/B. Node P2 is valid for (indexes) only the model "rect" in the graph of models. A distinguished subgraph is set up for "rect" in the scene graph with P2/B assigned to it. No reasons are given for the association at this stage. This association serves as a starting point for the scene graph.

Next, we might look at P2/C. Once again, the only model applicable is "rect". However, we do not now simply assign P2/C to the existing subgraph for "rect", but require first that it be consistent with earlier assignments - in this case with P2/B.

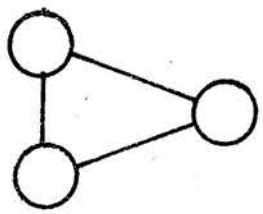
For consistency, some relation must be found between P2/C and P2/B with an acceptable value. Surfaces B and C are at 90°, and there is a relation schema, <RELANGLE P2 P2 90°>, that indexes "rect". Therefore P2/B and P2/C are compatible, and P2/C can be assigned to the subgraph for "rect". An arc joins the two nodes in the scene graph, labelled with the instantiated relation schema.

	pair	reasons
"rect"	P2/B	<RELANGLE B C 90°>
	P2/C	<RELANGLE B C 90°>



We next consider P1/A. Node P1 indexes both "cube" and "rect". Once again, we make no assignments without reasons. P1/A is found to be compatible with P2/B and P2/C in the subgraph for "rect", because A is at 90° to B and to C in the scene, satisfying the relation schema <RELANGLE P1 P2 90°> in each case. P1/A is therefore added to the subgraph for "rect", and the reasons of all assigned pairs are updated.

	pair	reasons
"rect"	P2/B	<RELANGLE B C 90°>
		<RELANGLE A B 90°>
	P2/C	<RELANGLE B C 90°>
		<RELANGLE A C 90°>
	P1/A	<RELANGLE A B 90°>
		<RELANGLE A C 90°>



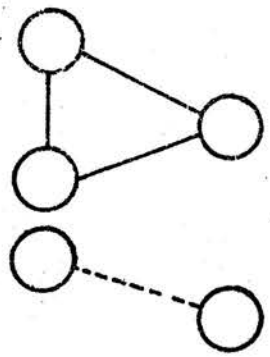
P1/A does not have any reasons to be assigned to a subgraph for "cube" because there are no previous assignments for it to be related to. Because it has found reasons to be assigned to the subgraph for "rect", no subgraph for "cube" is set up.

We have now unambiguously identified the block ABC as an instance of "rect". There are still four pairs in SP not assigned, namely P1/D, P1/F, P2/F, P2/E. Suppose we next choose P1/D to work on. This gives rise to a new situation. Node P1 indexes both the models for "cube" and for "rect". There is still no subgraph for "cube" in the scene graph, while that for "rect" has three assigned pairs. When relations between P1/D and the pairs assigned to "rect" are tested, none of them have suitable values. Thus there are no reasons at this stage for assigning P1/D to the subgraph for "rect".

In this circumstance, P1/D will be assigned to two new subgraphs in the scene graph. A second subgraph for "rect", called "rect1" will be created, and P1/D will be assigned to it, as well as to a subgraph for "cube". There are no reasons for the assignments at this stage.

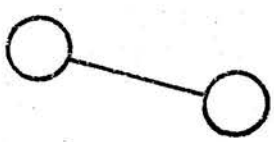
The situation now looks like this:

	pair	reasons
"rect"	P2/B	<RELANGLE B C 90°>
		<RELANGLE A B 90°>
	P2/C	<RELANGLE B C 90°>
		<RELANGLE A C 90°>
	P1/A	<RELANGLE A B 90°>
		<RELANGLE A C 90°>
"cube"	P1/D	
"rect1"	P1/D	



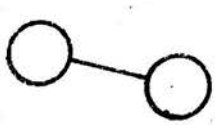
P1/F comes next. Again P1 indexes both "cube" and "rect". Since F is at 90° to D, the schema <RELANGLE P1 P1 90°> succeeds. This schema is valid for the model of the cube, and is the reason for adding P1/F to the subgraph for "cube". No relations valid for "rect" succeed, however, so P1/F is not added to "rect".

	pair	reasons
"cube"	P1/D	<RELANGLE D F 90°>
	P1/F	<RELANGLE D F 90°>



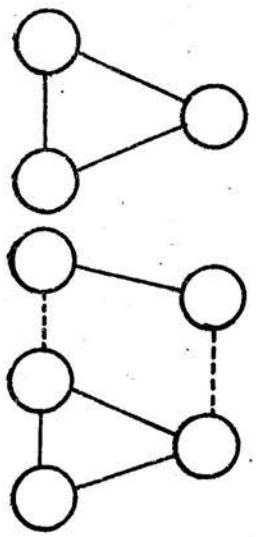
The reverse situation applies when the pair P2/F, linked to P1/F is encountered. Now the schema <RELANGLE P1 P2 90°> succeeds for "rect1", but nothing succeeds for "cube". Therefore, P2/F is added to "rect1".

	pair	reasons
"rect1"	P1/D	<RELANGLE D F 90°>
	P2/F	<RELANGLE D F 90°>



Finally, P2/E indexes only "rect", and of the two subgraphs for "rect" in the scene graph, only the second, "rect1", has assignments that relate correctly to P2/E. P2/E is thus assigned to "rect1", giving the assignments after all pairs have been processed as follows:

	pair	reasons
"rect"	P2/B	<RELANGLE B C 90°>
		<RELANGLE A B 90°>
	P2/C	<RELANGLE B C 90°>
		<RELANGLE A C 90°>
	P1/A	<RELANGLE A B 90°>
		<RELANGLE A C 90°>
"cube"	P1/D	<RELANGLE D F 90°>
	P1/F	<RELANGLE D F 90°>
"rect1"	P1/D	<RELANGLE D F 90°>
		<RELANGLE D E 90°>
	P2/F	<RELANGLE D F 90°>
		<RELANGLE E F 90°>
	P2/E	<RELANGLE D E 90°>
		<RELANGLE E F 90°>



There are two sources of ambiguity here. The pair P1/D is assigned to two subgraphs, and there are linked pairs P1/F and P2/F resulting from the occlusion of surface F. A constraint analysis algorithm, based on the reasons for assignment, is used to disambiguate the interpretations.

P1/D is in "cube" and in "rect1". However, there are better reasons for P1/D to be in "rect1", because more relations have succeeded there. The node P1/D is thus deleted from the subgraph for "cube", together with the arc joining P1/D to P1/F and its label.

But now P1/F no longer has reasons for being in the subgraph for "cube". Because P1/F is linked to P2/F, and P2/F has good reasons for being assigned to "rect1", we can delete P1/F from the subgraph for "cube". This means that the subgraph for "cube" no longer has any assigned pairs, and can be deleted.

There are no more ambiguities, and the final result is: "rect" is composed of P1/A, P2/B, and P2/C, and "rect1" is composed of P1/D, P2/E, and P2/F. That is, there are two copies of the object known as "rect" in the scene, described by the graph of Figure 6.2.

The sequence of choosing pairs used here was chosen to allow the

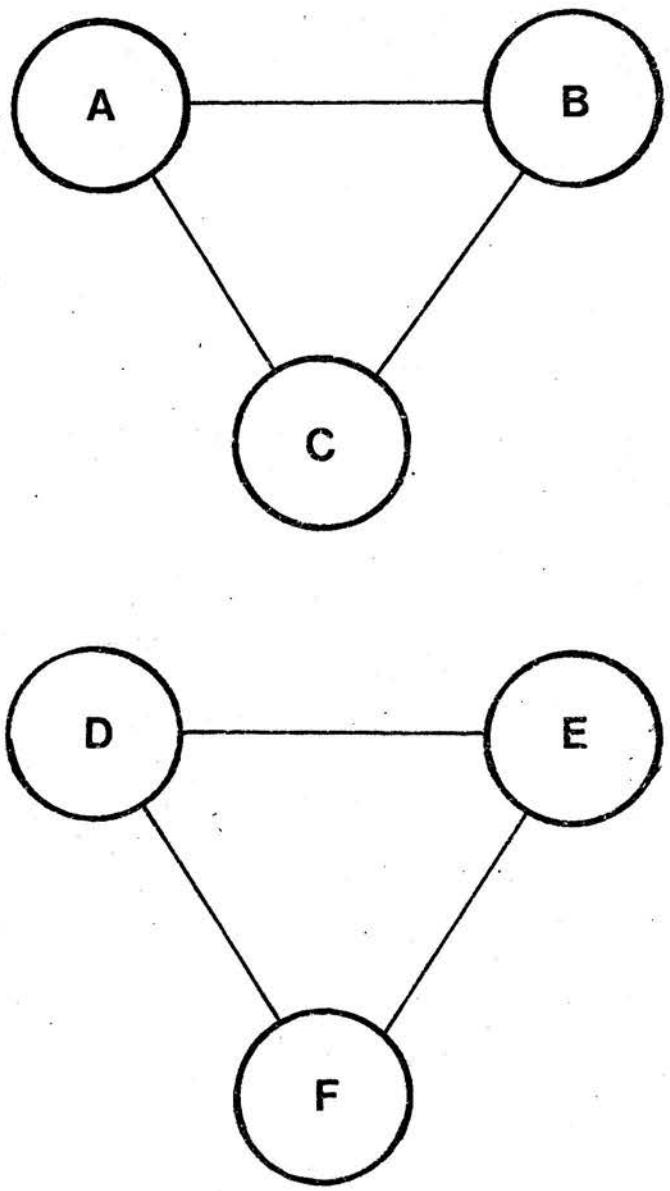


Figure 6.2

disambiguation step to be demonstrated. It also provides a clear demonstration of the fact that order is important to the efficiency of the algorithm. The sequence of choices of pairs is more efficient when the system that was implemented analyzes this scene. The heuristic of first choosing the pairs that index only one model directly and unambiguously identifies the scene.

6.3 THE RECOGNITION ALGORITHM.

The recognition algorithm first constructs a scene graph containing all nodes and arcs for which it can find any justification (steps 1 to 5). This scene graph is then examined in step 6. Multiple nodes corresponding to a single surface are sources of ambiguity. The reasons, or relations on the arcs leading from these nodes, are used as the basis for disambiguation. A node will be deleted if it has a rival that is better substantiated. The arcs leading from the node are also deleted, thus reducing the justification of the nodes attached to the other end of the arcs. This process continues until no more nodes can be removed. The remaining scene graph constitutes the interpretation of the scene.

The algorithm is in 6 steps.

1 The scene is scanned to form surface descriptions, a note being made of which surfaces are occluded or shadowed. The surfaces are matched against primitives in the database. A single surface may match and be paired with several primitives, in which case the primitives are called linked. Occluded and shadowed surfaces match all compatible surfaces larger than themselves.

The surface information is paired with each matching

primitive because it will be needed for working out the relations. The primitives indicate what models are relevant to the surfaces.

2 Assign those primitive/surface pairs whose primitives index only one object model to distinguished subgraphs of the scene graph associated with that model. If two assignments to the same subgraph are inconsistent the program assumes that there are two distinct objects with the same model in the scene, and two subgraphs for that model are set up. The tests for consistency are carried out as described in Step 5, by testing the relationships between the surfaces. These initial assignments have zero confidence.

3 While there are still unassigned primitive/surface pairs, repeat steps 4 and 5.

4 Choose one of the primitive/surface pairs which have yet to receive an interpretation. This choice can be important to the efficiency (but not the effectiveness) of the algorithm. It must be made carefully. The model associated with the subgraph of the scene graph that has most pairs assigned to it is chosen. A primitive/surface pair relevant to that model is sought, should such a pair exist. Otherwise, the pair whose primitive best matched with its surface in step 1 is chosen.

5 Test the relation constraints in the following way. The new primitive indexes a set of models. Each of these indexes a set of relations that the surface should satisfy if it does come from an instance of that model. A relation schema will be

evaluated if a subgraph for an instance of the model exists and has pairs assigned to it. The primitives of the pair must be of the same type as those of the schema's arguments. The surface part of the new pair will form one of the arguments for the relation test. The other argument will be the surface associated with the pair already assigned to the subgraph. The value of the test is the result of applying the function to the surfaces associated with the primitives. This value is compared with the expected outcome in the model. For a success, it must match to within a fixed tolerance (currently about 20%). A relation schema may be valid for several models. When a test using the schema succeeds it enhances the confidence that its arguments come from instances of each one of these models. Thus a single relation test can cause assignment of pairs to several subgraphs. If necessary a new subgraph will be created for a model, and nodes assigned to it for each argument of the relation schema. Failure of a relation test reduces confidence only in the suitability of the new primitive.

If no relation tests succeed, the primitive/surface pair is assigned to new subgraphs for all models indexed by the primitive, with zero confidence (i.e. with no reasons). This ensures that there is at least one node in the scene graph for each pair.

6 Apply filtering to the subgraphs constructed in the previous steps. An initial confidence in each assignment has been worked out, this being the set of reasons for assigning the pair to the subgraph. A way is sought to reduce the number of interpretations of each surface consistently on the basis of

these confidences. Note that a surface may be assigned to several subgraphs (i.e. have several interpretations) for two reasons. A surface may match with several primitives, forming more than one pair to be interpreted. Also, each pair may be assigned to several different subgraphs, gaining more than one interpretation for the surface associated with the pair. Any unique interpretation is retained, as is any whose rivals all have zero confidence.

When all interpretations have reasons, the situation is not so clear. If one interpretation has better reasons than any of its rivals, it is retained, and its rivals are deleted. Where the reasons are nearly equal, this is not justified, and all interpretations are considered as ambiguous identifications, pending further analysis.

Deleting a primitive/surface pair from a subgraph affects the confidence of all pairs related to it. When the pair is deleted, all arcs leading from it to other pairs are deleted together with the relations that label the arcs. Thus, the reasons for the related pairs are reduced, the amount of reduction being exactly the amount by which the deleted pair contributed to their justification. This is the way in which the filtering is accomplished, confidence reductions causing further deletions elsewhere, until no further changes are possible.

Linked primitives are treated as multiple interpretations of the same pair for the purpose of conflict analysis. Pairs are deleted if a better match is found with one of their links.

The process cycles while it is still possible to adjudicate

between assignments, there being a smaller number of nodes in the scene graph at each step. When all possible deletions have been made, the subgraphs remaining are assumed to describe the objects in the scene.

6.4 DISCUSSION OF THE RECOGNITION ALGORITHM.

The first step in the algorithm is to index the graph of models. A surface will match all primitives that have dimensions similar to those of the surface, and are of the same type. An assumption is made that, amongst these matches, one is correct. A number of primitives are allowed to match with a single surface, so it is necessary to keep track of a number of different instantiations. This is the function of the linking mechanism, which ensures that all the primitive/surface matches are treated uniformly.

Having found the primitives that match the surfaces, the next stage is to find assignments of primitive/surface pairs to subgraphs corresponding to instances of models. The purpose of this process is twofold. Firstly, a correspondence is established between parts of the scene and models in the graph of models. This correspondence is not one-to-one because a part of the scene may be interpreted as belonging to several different objects. The second purpose of the process is to estimate the confidence in each interpretation. These estimates form <u>a priori</u> confidences for the constraint analysis.

If a primitive indexes only one model, then a pair involving the primitive can be assigned to a subgraph for the model as soon as the pair is encountered.

Two primitives might index the same model, or two surfaces matching a particular primitive might appear in the scene. In this

case, before assigning the pairs to a subgraph for the model, they are tested for compatibility. Each model indexes a set of relation schemata, some of which have arguments of the same type as the two primitives. Compatibility is tested by applying the function part of the schema to the surfaces paired with the primitives. Two pairs are compatible if the result of the function application is similar enough to the value expected by the schema. Should this be the case, both pairs are assigned to the same subgraph. The reasons given for the assignments are the instantiated relation schemata that indicated the compatibility. These instantiated schemata label the arc that is created to join the pairs. If no relations succeed, the surfaces are assumed to be from different instances of the same model. A subgraph for the model is set up for each pair and the pairs are assigned to different subgraphs.

The rest of the primitive/surface pairs are assigned to subgraphs in a similar manner. A pair is chosen, and all models that the primitive part of the pair indexes are found. Different primitives will index different models and at any time, only some of these models may have subgraphs associated with them in the scene graph. It is important for the efficiency of the algorithm to choose the pairs in an order that results in the smallest possible set of subgraphs being constructed. The heuristic used to make the choice is to find the subgraph that currently has the most pairs assigned to it, or in which the pairs have greatest justification. Should there be a primitive/surface pair still unassigned whose primitive indexes the model associated with the subgraph, that pair is chosen as the next to be assigned. Otherwise, the pair is chosen whose surface best matched with its corresponding primitive. This heuristic works

well in practice, because it only increases the number of subgraphs when it has no other choice. The order in which the pairs are chosen does not affect the final result of the recognition.

Note that, although a particular model might have suggested the choice of a pair, the subgraphs associated with that model have no special claim to the pair. The assignments are made on the basis of relation tests. Each model indexed by the primitive contributes relations that will decide the assignments. A relation may be applied if all its arguments can be instantiated. If a relation succeeds, the new primitive/surface pair is assigned to subgraphs for all models indexed by that relation schema.

Usually the models indicated by the relation schema will already have subgraphs associated with them. It may happen that a model that was previously uninstantiated is indexed by a relation schema. In this case, a subgraph will be created for that model and both pairs that caused the relation to succeed will be assigned to the subgraph. An arc will be created to join them, labelled by the instantiated relation schema.

The first assignment to a subgraph will have null reasons (or zero confidence). Thereafter, assignments can only be made if a relation test succeeds. The reasons consist of instantiated relations, whose primitives have been replaced by their corresponding surfaces. Reasons are updated after each successful relation test.

When no relation tests involving a new pair succeed, the pair is assumed to belong to a previously unknown object in the scene. (e.g. the assignment of P1/D in the example of section 6.2.1). It is thus assigned to subgraphs for all models indexed by its primitive. If necessary, new subgraphs for models that were already instantiated

are created. The reasons given for the assignments are null.

The assignment of pairs to subgraphs continues until all pairs have been assigned. At this stage there are often multiple possibilities for the identity of each surface. Occlusion and shadowing may give rise to linked primitives (e.g. surface F in the example of section 6.2.1). Surfaces may also have been assigned to several subgraphs in the scene graph (e.g. surface D in the example).

The primitive parts of the pairs have now served their purpose, and are no longer considered important in finding the final interpretation of the scene. They correspond to the nodes in the intermediate structure obtained from the graph of models. Now that the scene graph has been constructed, we want to find that part of it that gives the best interpretation for each surface. There is enough information in the scene graph to enable this to be done without further reference to the intermediate structure or the graph of models.

Each assignment to a subgraph has reasons. The reasons are used to guide a constraint analysis algorithm that tries to reduce the number of interpretations of each surface. If a surface has a unique interpretation, that interpretation is assumed correct, because there are no alternatives. If all interpretations have no reasons, then all the interpretations are kept, because there is no way of distinguishing between them (although they can be ranked according to how well they matched with their primitives).

The more interesting cases occur when all the alternative interpretations of surfaces do have reasons. If one interpretation has more reasons than its rivals, it is retained and its rivals are discarded. Where the reasons are all similar, the subgraphs are kept

as ambiguous interpretations, pending later resolution.

Discarding an interpretation is equivalent to removing a node from the scene graph. When this is done, all arcs attached to that node must be deleted, and their labels discarded. The result of this procedure is to reduce the reasons for the interpretation at the other end of the arc. Reducing the reasons for an interpretation might cause this interpretation to be rejected later.

The process cycles until it is no longer able to discriminate between interpretations. At this stage, the remaining interpretations are considered to identify the scene.

Only one branch of the solution is followed in the algorithm. The consequences of retaining an interpretation have not been investigated. It might happen that some interpretation might be deleted because it is less well supported than a rival. Later, however, the reasons for the rival interpretation might be eroded, so that the original interpretation is better. However, the first part of the algorithm has constructed a good a priori set of reasons for each interpretation. This justifies not considering the other branch of the search tree.

It may happen that a surface does not match exactly with any one primitive. This could be caused by occlusion or shadowing, or because of errors in measurement. In such cases, the surface is matched with all primitives larger than itself. The primitives are linked together. The links are such that if the interpretation of one pair is better than that of pairs linked to it, the rival pairs can be discarded. This is done in the same way as the disambiguation of rival interpretations of a single pair.

The recognition algorithm is guaranteed to terminate: In the

first stage, the cycle of steps 3-5 always reduces the number of unassigned pairs. When this number reaches zero, the filtering stage is entered. Here, once again, the cycle has to remove an interpretation to enable filtering to proceed. There are only a finite number of assignments, so termination is guaranteed.

The algorithm is not guaranteed to give the correct result for two reasons. The first is that the initial match between surfaces and primitives might not include the correct primitive. An extension to the algorithm has been implemented to allow rematching with relaxed criteria (see section 6.4.1).

The second possible source of error is the incomplete search in the filtering, mentioned above. This source of error could be removed by performing the extra search indicated.

6.4.1 EXTENSIONS TO THE ALGORITHM.

Two extensions to the algorithm have been implemented. Firstly, a method of relaxing the constraints for matching surfaces to primitives has been provided. The recognition algorithm might fail to match a surface with the correct primitive. This might mean that the final result is unsatisfactory. When the algorithm has terminated, the user is given the opportunity to initiate further effort. The set of interpretations is then examined in an attempt to find an interpretation that is supported by a single surface. That surface is matched again with the primitives, using relaxed constraints on the match. If any further successful matches are found, the recognition algorithm resumes, using the new pairs. The hope is that the initial match was incorrect. If the new match is correct, the previous interpretation will be discarded, and the new

interpretation will take its place. An example of this process is given in Chapter 7.

This process could be automated. The system could always re-examine all interpretations supported by only one surface. Using the range data, however, it happens reasonably often that only one surface of an object is visible in a scene. It would be expensive to always try an extra search at this stage before allowing the surface to contribute to the analysis of the scene, so the matching was not automated in the implementation.

The second extension comes into play at the end of the conflict analysis. It makes use of the expectation that the assignment of a particular interpretation to a large number of surfaces should give rise to a correspondingly large number of reasons for that interpretation. This implies that if there are equally good reasons for two interpretations of a particular surface, the interpretation supported by the smaller number of other surfaces should be retained. The interpretation with the larger number of surfaces should be discarded.

The reasoning behind this is that with a larger number of pairs assigned to a subgraph, there should have been better reasons (more successful relation tests) for the assignment. Because there are not, it must mean that some relation tests failed when they should have succeeded. On the other hand, with a smaller number of assignments, the number of unsuccessful relation tests must have been smaller, or the confidence would have been less.

This heuristic depends on a number of factors that may not hold in general. It assumes that the weighting function that assesses the value of relations in the conflict analysis treated both instances

alike. This in turn assumes that the models were defined in terms of commensurate relations, and that the relations were defined uniformly between all primitives of the models.

The extension opens up the question of the use of negative information. The fact that some expected result did not occur can be of great significance in an interpretation. The conclusions that can be drawn depend strongly on the kind of relations, though, and it is hard to conceive of a uniform technique for handling this kind of information.

6.4.2 ALTERNATIVE ALGORITHMS.

The recognition algorithm described here is only one of a number of similar procedures. Its form was dictated in part by the input mechanism. The only interface between the imaging and interpreting stages of the system is through disc files. It is necessary that a complete scene be scanned before recognition can be attempted. This means that all the information is available at the start of the recognition, and there is no chance of directing the image-taking process by means of expectations derived from the recognition.

Two modified versions of the algorithm have been produced. The first of these differs from the original algorithm in the matter of what relations may be tested when a primitive/surface pair is first assigned. Normally only the models that have already been associated with a subgraph may contribute relations for compatibility tests. In the alternative approach, each primitive indexes the set of relations in which it appears as an argument. Any of these, whose other argument already has an instance assigned to <u>any</u> subgraph, may be applied when assigning the new pair.

With the normal method of choosing relevant schemata, there is the chance that some models might never be instantiated, even though they are candidates for assignment. This is because a relationship that holds between two surfaces might not be expected by models that have already been instantiated, and so will not be tested for. The alternative algorithm removes this possibility. Usually, however, it is more costly than the normal method because it applies more relations and instantiates more models than are necessary for making the assignments.

The second alteration to the algorithm involves removing the heuristics for choosing the best pair for assignment. This does not affect the ultimate result of the recognition, but does cause a lack of direction in the search. There is also a definite possibility of extra work being done, due to poor sequencing of pair assignments. This is because pairs might have to be assigned to more subgraphs using one sequencing than they would with another, and this could result in extra relations being tested. In practice, however, it was found that removing the heuristics had little effect on the course of the algorithm. With a larger domain of objects, the heuristics would have a bigger part to play.

By changing the relationship between the image-taking and recognition stages, other algorithms become more feasible. A very close interaction allows the possibility of dynamic verification. Having scanned enough of the scene to obtain a complete surface, the matching of the surface with the primitives will suggest some models to describe the object in the scene. These models can provide information about where to search and what to search for, to interpret the scene most efficiently. This is analogous to the work

of Grape(1973) in object recognition, and of Shirai(1975) in line-finding.

6.5 DISCUSSION

The greatest benefits of using the compact representation occur in recognition. Advantages are the ease with which models are accessed, the discrimination over which tests to apply to distinguish between interpretations, and the treatment of symmetric objects.

All models have been distributed in a single network and index paths into the network have been provided based both on surface descriptions and on relationships between surfaces. When a part of the structure is accessed through one of the index paths, all models that are concerned with that part of the structure are made available simultaneously. There is no need to match successively into different structures for each possible model (as did Winston(1975), Barrow and Popplestone(1971), for example), nor to access a central index that points to the relevant models (e.g. Grape(1973)). The program does the minimum amount of work to gain the maximum amount of information.

When the graph of models is indexed, the part of the structure that is indexed is also made available. This part of the structure can be examined, as can all parts that are linked to it. Usually the links between primitives, and the relation schemata that label the links are of special interest. The program can choose from among the sets of schemata those that best serve its current interests. Thus, if the program is trying to discriminate between two interpretations for a surface, it looks for schemata that are relevant to those two interpretations, and ignores any others that may be applicable. This

discriminatory ability enhances the efficiency of the recognition process because it is not necessary to apply all possible tests before making a decision. The ability to use knowledge selectively is important, especially when there are many possible tests, only a few of which could be useful.

The structure that is built to describe a scene has two functions. The first of these is as a scene graph, that is, a description of the scene in terms of parts of the scene, relations between the parts, and interpretations for each of the parts. The second purpose of the structure is to instantiate parts of the graph of models. The interpretation of this aspect of the structure is as a particular set of models against which the parts of the scene are to be matched. That is, the graph of models is a structure that can represent a number of different conventional models structures. The particular configuration chosen depends on the scene being studied.

Thus the recognition algorithm has two tasks to accomplish. It must decide on the nature of the intermediate structure to be constructed from the graph of models, and it must interpret parts of the scene as corresponding to parts of the structure. The goal of relational structure matching is to find a one-to-one correspondence between parts of the scene and parts of the model. By constructing a single integrated structure for both the parts of the scene and the particular model configuration, we accomplish the matching automatically.

The process can also be looked on as the construction of a template to fit the scene, under the guidance of the graph of models. The matching is then a many-one process, with several parts of the scene mapping to the same part of the graph of models. In either

case, the structure that is constructed reflects the characteristics of the particular scene being analyzed.

A notable consequence of this feature of the structure is that symmetric objects are matched with a tailor-made structure that makes such matches unambiguous. The problem of finding which part of the model to match with which part of the scene disappears.

There are two main problems in using the recognition system. The first stage of the process involves building a redundant structure to represent all interpretations of a scene. This structure may grow large if each part of the scene has several local interpretations. The whole of the structure is currently required to exist before the constraint analysis is applied. It would be useful if some of the constraint analysis could be interspersed with the construction of the scene graph. The problem with applying the constraint analyzer early is that interpretations may not have attained their full justification, causing the wrong interpretations to be deleted.

In order to be able to mix interpretation with constraint analysis, it seems that extra knowledge must be applied. Some of this knowledge may not be teachable using the modelling system described in Chapter 5. Thus, one would need to know about the space occupied by objects to enable the constraint analyzer to be called when a particular volume of space was filled. Another piece of necessary information would be the physical constraint that no two objects can occupy the same space at the same time. Such information would supply both extra global constraints and indicate when it was safe to call the constraint analyzer.

Another problem arises from the ability to describe different

objects by means of different relations. When the constraint analyzer tries to distinguish between two interpretations of a part of a scene, it must be able to compare the reasons for each interpretation. If both interpretations are justified by the same sort of relations (e.g. if both surfaces have ADJACENT links to other surfaces) the comparison is easy. However, allowance has to be made for the case where the relations are not the same. There must be some way of ranking the importance of different relations and their contribution to an interpretation. In the implementation a weighting function was used to rank relations. Various rankings were tried. In small domains with only four or five relations, the ranking did not seem to be very important. In a larger system or one in which some relations were much more reliable than others, a good set of weights would be essential. It would be much more satisfactory to have some metric on relations that reflected their contribution to an interpretation directly. The weighting that was finally chosen is based on the number of objects indexed by a relation schema. The weight of a relation is the reciprocal of the number of objects it indexes. Thus, certain relations have a high value because they usually give rise to reliable identifications, while others are less important because they are less useful for disambiguation. A ranking scheme based less on the specific set of objects that have been modelled would make both the modelling and the recognition easier. When modelling, relations that had a high ranking would be used. When performing recognition, the ranking would provide a reliable comparison between relations.

The domain to which the system is applied has a big effect on the kind of relations that are important. In the blocks world the

relation of perpendicularity is not very useful because of the number of objects whose surfaces are perpendicular to each other and the likelihood that surfaces from different objects will coincidentally be perpendicular. In a domain where most objects do not have perpendicular surfaces, such as the Barrow and Popplestone world, the existence of a perpendicular pair of surfaces can be most enlightening.

A number of people have studied the properties of different kinds of constraint analysis algorithms. Mackworth(1977) described a constraint analysis problem by means of a network consisting of a labelled directed graph whose nodes were variables, each with an associated set of possible values, and whose arcs corresponded to predicates. The problem was to find the set of values for the nodes that satisfied all the predicates. He isolated three sources of potential inefficiency in operating on these networks, called node, arc, and path consistency.

Node consistency refers to the requirement that all values for a variable at a node satisfy the unary predicates for that node. This is easy to achieve, by discarding those values that are illegal. Doing this before any other processing saves a great deal of unnecessary effort. In the algorithm presented in this thesis, the node consistency problem is solved by only attaching consistent interpretations to nodes.

The second problem, that of arc consistency, concerns binary predicates, and requires that the values at pairs of nodes satisfy the predicates defined between the nodes. That is, if some value in one node does not have a corresponding value at the other node that would make the predicate true, then the value at the first node may

be discarded. This is the basis of the Waltz algorithm. The algorithm in this thesis ensures arc consistency by making a queue of all arcs in the network, applying a local operation to each arc sequentially, making the nodes consistent, and re-entering on the queue any arcs that may be affected by the changes, if they are not already there. This is similar to the third arc consistency algorithm described by Mackworth.

The third source of inefficiency, path consistency, refers to the situation in which node and arc consistency have been achieved, but where values assigned simultaneously to variables at two nodes cause a third node to have no acceptable value. Obtaining the constraints from the models ensures that this situation does not occur in the current work.

Rosenfeld et al.(1976) developed mathematical models for various versions of constraint satisfaction algorithms. These algorithms ranged from the discrete labelling of the kind used by Waltz, to probabilistic models. The model most similar to that discussed in this thesis involved the assignment of weights to the interpretations. Rosenfeld et al. showed that if the maximum weight was initially assigned to all interpretations and these weights were successively weakened on the basis of the compatibility relations, then the result was the set of interpretations with the strongest weights.

Interpretations in the present work have confidences or weights as a result of surfaces being interpreted as belonging to model instances. The weight of interpretations can only decrease, and never increase. In these respects, the algorithm is similar to the weighted model of Rosenfeld et al. There are, however, marked

differences. For example, the compatibility relations that determine the reasons for interpretations are not defined for all pairs (or n-tuples) of labels, but only for those that have been determined to interact. This means that only relevant information can affect the weight of a label. The current method also allows models to direct the assignment of weights, resulting in further efficiency.

The current work is an advance on earlier work like that of Barrow and Tenenbaum(1976) in the initialization of the constraint analysis. In the past it has been necessary to provide the set of interpretations and their weights for each part of a scene being analyzed. In this work the initialization is achieved by accessing the graph of models and constructing the interpretations and constraints automatically.

A comparison can be made between the indexing in this work and the indexing schemes discussed by Marr and Nishihara(1977). Marr and Nishihara describe three-dimensional objects in terms of stick-figure configurations. Each stick represents one or more axes in the object's generalized cylinder representation. Models are hierarchical, the hierarchy being based on the degree of detail. For example, at the top level, a human model is described by a single cylinder giving rough size and orientation. This can be broken down into components for the torso, head, and limbs, each of which is described similarly. The decomposition can be continued to any desired depth. The axes of components are referred to the coordinate system of the model above them in the hierarchy (called an adjunct relationship). This is not the same as the conventional hierarchical decomposition scheme, in which parts are put together to make an object in a rigid fashion. In this system, levels could be skipped,

or ignored, without seriously degrading higher levels. This sort of hierarchy is useful for ensuring the conservatism of the representation.

Three kinds of indexing schemes are described for recognition. The first makes use of the different precision of different levels of the hierarchy, and is called the specificity index. A newly-derived model may be related to the set of stored models by starting at the top, and working down until the level of specificity corresponds to that of the new model. Another access path is called the adjunct index. Once a three-dimensional model has been selected, the relations between its axes and those of its components allow the components to be accessed based on their locations, orientations, and relative sizes.

The third access path is known as the parent index. When a component of a three-dimensional model is recognized, it can provide information about what the whole shape might be. For example, a horse's head indicates not only that the horse model is appropriate, but can constrain the location of the rest of the horse.

The adjunct and parent indices play a secondary role to that of the specificity index in the scheme of Marr and Nishihara, providing constraints to support the derivation process.

The main indexing in the present system corresponds to Marr and Nishihara's parent indexing, with more emphasis on constraint satisfaction to reduce the number of possible interpretations. Generality and "graceful degradation" are handled in Marr and Nishihara's system by the specificity index. The system presented in this thesis accesses all models compatible with the data, and then weeds out those that are inappropriate.

The recognition can be seen to retain most of the advantages of earlier work. It has the added power to deal effectively with symmetry, and to assign interpretations and weights for constraint analysis automatically.

We have now examined both the representation and the recognition in detail. The next chapter presents some examples of the application of the system to real scenes. Chapter 8 illustrates a further application in the domain of spelling correction.

7.0 EXAMPLES

This chapter illustrates the use of the modelling and recognition systems in the domain of machine vision. The objects in the domain have surfaces that are either planar or cylindrical. The data for the examples were acquired from the ranging device (Section 5.1). Objects were scanned against a background of dark cloth to ensure that only the objects themselves appeared in the scene.

There are two processes to be illustrated. The first is the construction of the model of an object from a set of scans. This process is not subject to a great deal of variation, and will be illustrated by one example. The model of a toy car will be constructed from four scans that together encompass the whole car. The second process is that of recognition. Each scene to be recognized poses its own problems. A number of examples of recognition will be presented, highlighting different aspects of the system.

7.1 MODELLING

When an object is to be modelled, it is placed on a turntable to enable a measured amount of rotation between scans. Figure 7.1 shows a picture of the toy car resting on the turntable. The car was

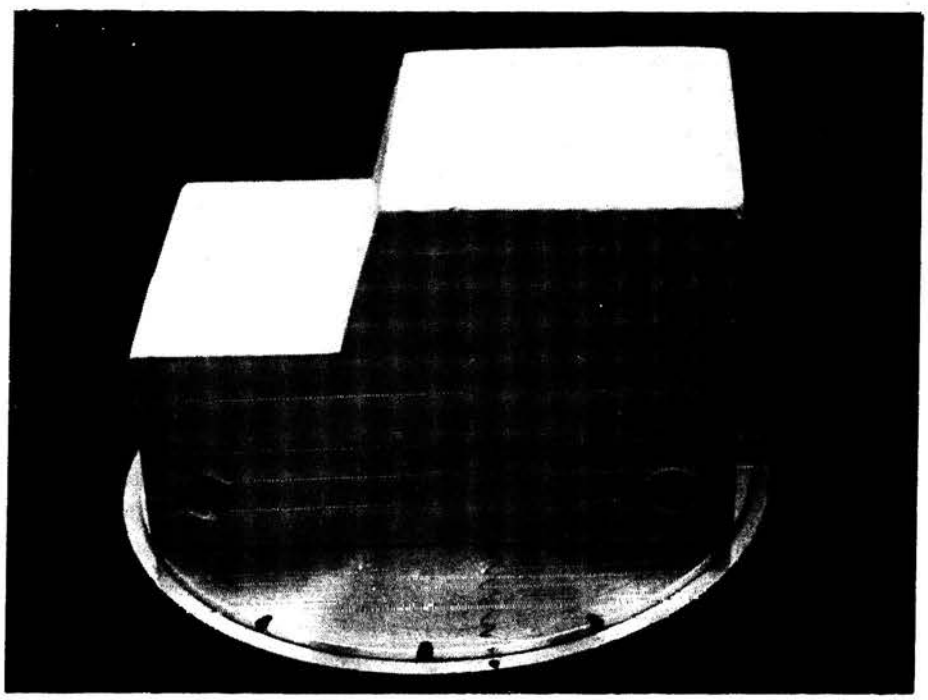


Figure 7.2

scanned by the ranger, rotated through 90 degrees, and scanned again. This process was repeated four times, giving four views, each perpendicular to its predecessor. The four views cover all surfaces of the car except its base (Figure 7.2).

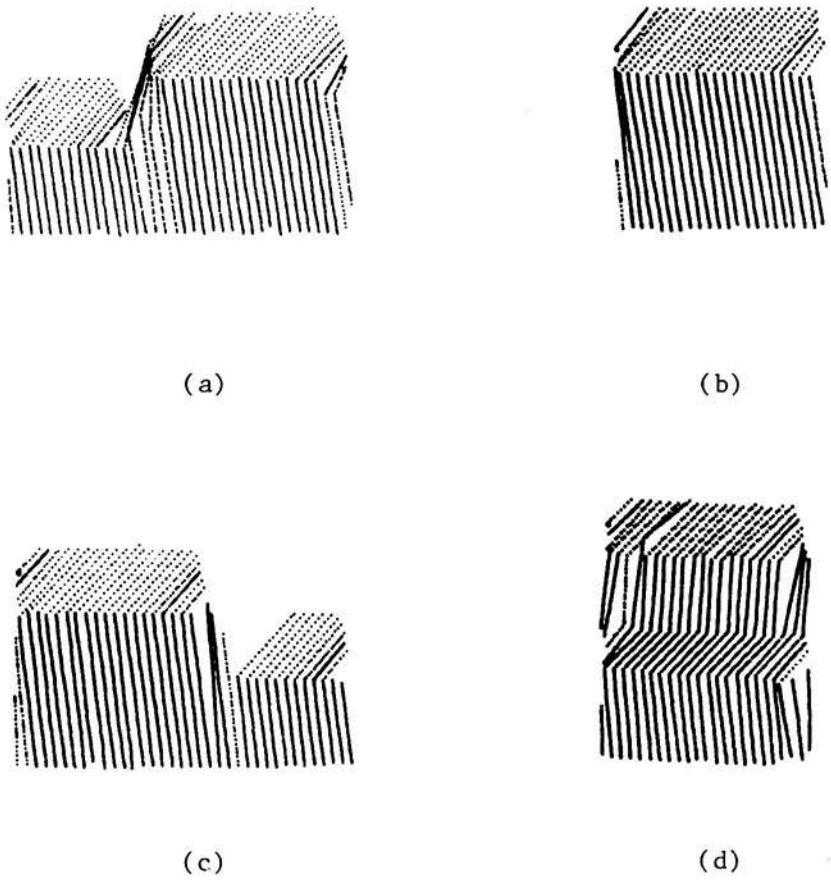
After scanning the car, the range map produced from the scans is processed to find the surfaces of the car. There are seven of these, corresponding to the roof, two sides, back, front, windshield, and bonnet of the car. Some of these surfaces were visible in more than one view of the car. The roof appeared in all four scans, and the bonnet in three (Figure 7.2). Notice that the sides of the car appear smooth because the axle holes were not large enough to be perceived by the ranger.

From the set of surfaces, the program must produce a description of the car. To be able to do this, three pieces of information are necessary, and are given as the three arguments to MODEL, below. The first is a name for the object - it will be called "CAR". The second requirement is a file containing the surface data, and the third requirement is a list of functions to be applied to pairs of surfaces to produce the relation schemata. We will use a single function that finds ADJACENT relationships between surfaces. Thus, to initiate the modelling process, the user types

```
MODEL("CAR",[CAR],[ADJACENT]);
```

The first thing that the program does is to work out descriptions for each surface. It finds the type of the surface (in this example, all surfaces are planes) and two numbers that describe the dimensions of the surface. All dimensions in this chapter are in metres.

When the program tries to find the dimensions of the roof of the



(a) (b)

(c) (d)

Figure 7.2

car (Surface S1), it discovers that there are several views of the surface, that is, the roof was scanned more than once by the ranger. Each of these views is processed separately, and the results are merged. The program signals a new view by printing NEW VIEW S1. The bonnet of the car is treated similarly. The result of finding the dimensions of all the surfaces is as follows:

```
SURFACE S1
NEW VIEW S1
NEW VIEW S1
NEW VIEW S1
 0.1595  0.1156
SURFACE S2
NEW VIEW S2
NEW VIEW S2
 0.1032  0.0729
SURFACE S3
 0.2457  0.1079
SURFACE S4
 0.2234  0.1206
SURFACE S7
 0.142  0.1351
SURFACE S8
 0.1247  0.0631
SURFACE S9
 0.1383  0.0817
```

Notice that the surface numbers are not all sequential. This is an artifact of the surface-finding program, which first finds surface fragments and then merges these into larger surfaces. The numbers for some of the fragments do not appear in the final set of surfaces.

Having described the surfaces, the next stage is to construct the relational structure to describe the object. First, the program examines the surface descriptions. If two descriptions are similar, they are represented by a single primitive node (PRIMITIVE) in the graph of models. In the example, three primitives are found to be sufficient to describe the seven surfaces. This is almost half the number that would be required by a conventional representation, illustrating the compactness of the representation. The three

primitives describe the following surfaces:

Primitive P1 describes the roof and back of the car.

Primitive P2 describes the windshield, bonnet, and front of the car.

Primitive P3 describes the two sides of the car.

The dimensions stored for each of the primitives are the means of the dimensions of each surface that matched with the primitive. In addition to describing the surfaces, the primitives also index the model CAR of which they are parts.

```
PLANE P1
BODYLIST CAR
EXTENT 0.1507  0.1254

PLANE  P2
BODYLIST CAR
EXTENT  0.1261  0.0749

PLANE  P3
BODYLIST CAR
EXTENT  0.2345  0.1143
```

The next step in building the model is to link the primitive nodes by means of ADJACENT relation schemata. An arc is constructed between two nodes if <u>any</u> of the surfaces described by one of the nodes is adjacent to <u>any</u> of the surfaces described by the other node. Thus, the side of the car is adjacent to the back of the car, giving rise to the schema <ADJACENT P1 P3 1>. The side of the car is also adjacent to the roof of the car, but this gives rise to the same schema, and one schema is used to represent both relationships. All pairs of surfaces are examined, and the final set of relation schemata produced is as follows:

```
ADJACENT  R5  ARGUMENTS  P2   P2
EXPECTED VALUE  1.0
BODYLIST CAR

ADJACENT  R3  ARGUMENTS  P1   P2
EXPECTED VALUE  1.0
BODYLIST CAR
```

```
ADJACENT  R1  ARGUMENTS  P1   P3
EXPECTED VALUE  1.0
BODYLIST CAR

ADJACENT  R2  ARGUMENTS  P1   P1
EXPECTED VALUE  1.0
BODYLIST CAR

ADJACENT  R4  ARGUMENTS  P2   P3
EXPECTED VALUE  1.0
BODYLIST CAR
```

The three primitives and five relation schemata, together with the name of the car, are all the information needed to model the car.

Note that the bottom of the car was not scanned, and so does not appear in the model. To incorporate it, two sets of scans would be necessary. The model as shown would first be constructed, and then another set of views of the car, this time on its side or its roof would be produced. Presenting the modeller with the new set of views and the old object name would result in the model for CAR being updated to include the new surface and any new relations that were discovered.

The situation is more complex when several objects are modelled. If the descriptions of some of the surfaces of an object being modelled match with primitives already in the graph of models, the new model will share that primitive in the graph. The node is made to index the new model in addition to any others it already indexes. Similarly, relation schemata may be shared by several models, and must be indexed accordingly.

For the purposes of the recognition examples to be presented, a database of six objects was created. These are the car discussed above, a pyramid, a cylinder, and three rectangular boxes. The boxes are so constructed that the model for each box shares a primitive with the models for each of the other boxes. As a result, the

relations become essential in unambiguously identifying any particular box.

The results of modelling all six objects are shown in Figure 7.3. Note that the graph of models is very compact - seven primitives and seventeen relation schemata are sufficient to describe all the objects. Most of the primitives are shared by several object models.

None of the relation schemata in these examples are shared, partly because of the choice of different relations to use to model different objects, but largely because the differences between the objects are captured by the relations. In the next chapter, examples will be shown of models that share both primitives and relations to a much greater extent.

Note that the pyramid was described by two primitives. The clustering routines found it hard to decide where to split the two faces visible in a view of the pyramid (see Figure 7.7b). As a result, one of the surfaces was perceived as slightly smaller and one as slightly larger than their actual size. This did not cause any confusion during recognition.

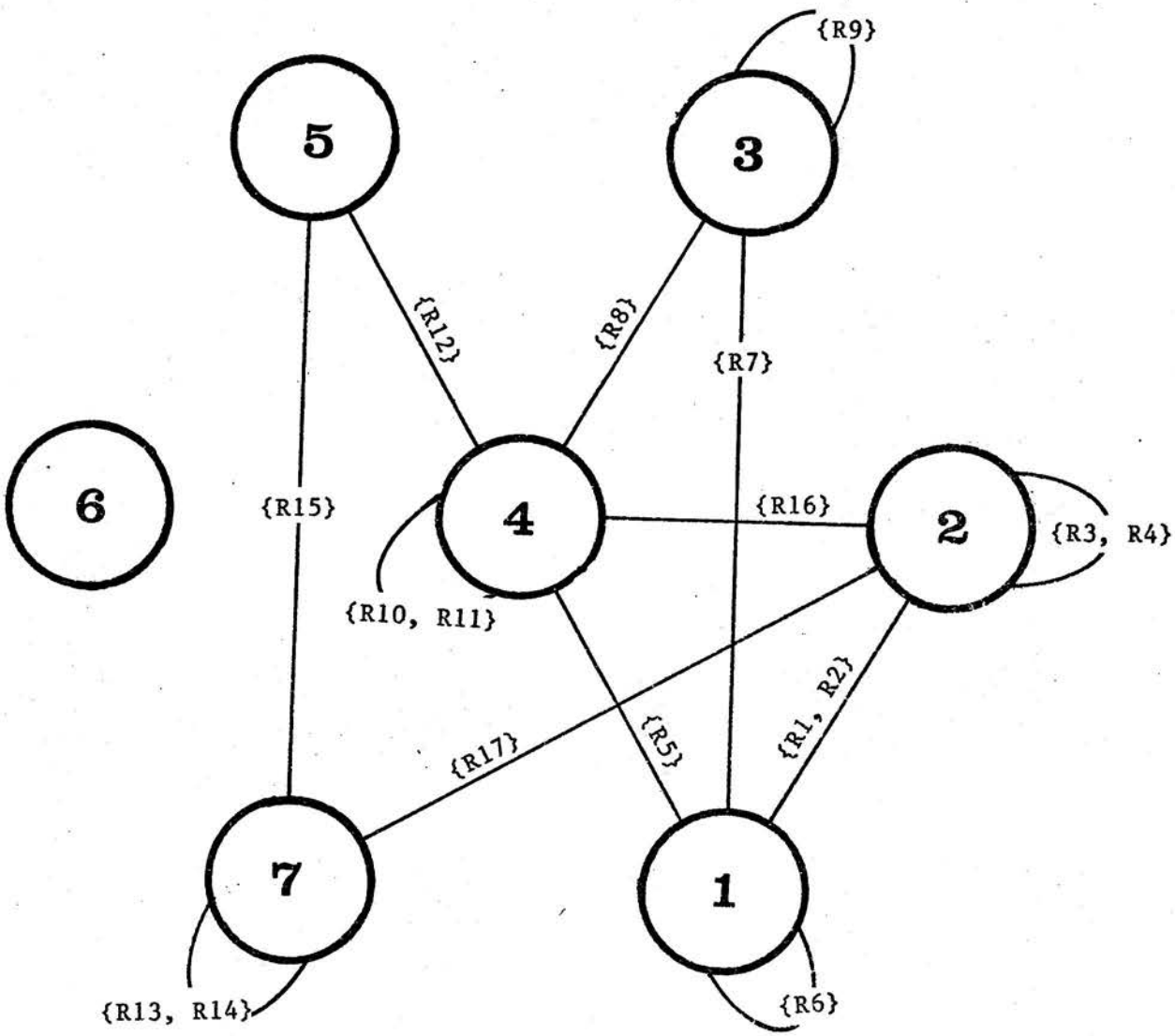


Figure 7.3

```
PRIMITIVES:


PLANE  P1
BODYLIST CAR PYRAMID
EXTENT  0.149  0.1281


PLANE  P2
BODYLIST BIGBOX PYRAMID
EXTENT  0.2241  0.1616


PLANE  P3
BODYLIST CAR
EXTENT  0.1261  0.0749


PLANE  P4
BODYLIST BIGBOX MEDBOX CAR
EXTENT  0.2345  0.097


PLANE  P5
BODYLIST SMALLBX MEDBOX
EXTENT  0.0984  0.0919


CYLINDER  P6
BODYLIST CYL
EXTENT  0.206  0.061


PLANE  P7
BODYLIST BIGBOX SMALLBX
EXTENT  0.149  0.0988
```

Figure 7.3(cont'd)

```
OBJECTS:

BIGBOX
DESCRIPTION
PLANE  P7
APPEARS IN RELATIONS: ADJACENT R17
PLANE  P4
APPEARS IN RELATIONS: ADJACENT R16
PLANE  P2
APPEARS IN RELATIONS: ADJACENT R17 ADJACENT R16

SMALLBX
DESCRIPTION
PLANE  P5
APPEARS IN RELATIONS: ADJACENT R15
PLANE  P7
APPEARS IN RELATIONS: ADJACENT R15 SAMENEXT R14 ADJACENT R13

CYL
DESCRIPTION
CYLINDER  P6
APPEARS IN RELATIONS:

MEDBOX
DESCRIPTION
PLANE  P5
APPEARS IN RELATIONS: ADJACENT R12
PLANE  P4
APPEARS IN RELATIONS: ADJACENT R12 SAMENEXT R11 ADJACENT R10

CAR
DESCRIPTION
PLANE  P4
APPEARS IN RELATIONS: ADJACENT R8 ADJACENT R5
PLANE  P3
APPEARS IN RELATIONS: ADJACENT R9 ADJACENT R8 ADJACENT R7
PLANE  P1
APPEARS IN RELATIONS: ADJACENT R7 ADJACENT R6 ADJACENT R5

PYRAMID
DESCRIPTION
PLANE  P2
APPEARS IN RELATIONS: RELDIST R4 RELDIST R3 RELDIST R2 RELDIST R1
PLANE  P1
APPEARS IN RELATIONS: RELDIST R2 RELDIST R1
```

Figure 7.3(cont'd)

```
RELATION SCHEMATA:

ADJACENT  R17  ARGUMENTS  P2   P7
EXPECTED VALUE  1.0
BODYLIST BIGBOX

ADJACENT  R15  ARGUMENTS  P7   P5
EXPECTED VALUE  1.0
BODYLIST SMALLBX

SAMENEXT  R11  ARGUMENTS  P4   P4
EXPECTED VALUE  1.0
BODYLIST MEDBOX

ADJACENT  R9  ARGUMENTS  P3   P3
EXPECTED VALUE  1.0
BODYLIST CAR

ADJACENT  R7  ARGUMENTS  P1   P3
EXPECTED VALUE  1
BODYLIST CAR

ADJACENT  R5  ARGUMENTS  P1   P4
EXPECTED VALUE  1.0
BODYLIST CAR

RELDIST  R4  ARGUMENTS  P2   P2
EXPECTED VALUE  0.1656
BODYLIST PYRAMID

RELDIST  R1  ARGUMENTS  P1   P2
EXPECTED VALUE  0.119
BODYLIST PYRAMID

RELDIST  R2  ARGUMENTS  P1   P2
EXPECTED VALUE  0.1746
BODYLIST PYRAMID

RELDIST  R3  ARGUMENTS  P2   P2
EXPECTED VALUE  0.1218
BODYLIST PYRAMID

ADJACENT  R6  ARGUMENTS  P1   P1
EXPECTED VALUE  1
BODYLIST CAR

ADJACENT  R8  ARGUMENTS  P3   P4
EXPECTED VALUE  1.0
BODYLIST CAR
```

Figure 7.3(cont'd)

```
ADJACENT  R10  ARGUMENTS  P4   P4
EXPECTED VALUE  1.0
BODYLIST MEDBOX

ADJACENT  R12  ARGUMENTS  P4   P5
EXPECTED VALUE  1.0
BODYLIST MEDBOX

SAMENEXT  R14  ARGUMENTS  P7   P7
EXPECTED VALUE  1.0
BODYLIST SMALLBX

ADJACENT  R13  ARGUMENTS  P7   P7
EXPECTED VALUE  1.0
BODYLIST SMALLBX

ADJACENT  R16  ARGUMENTS  P2   P4
EXPECTED VALUE  1.0
BODYLIST BIGBOX
```

Figure 7.3(cont'd)

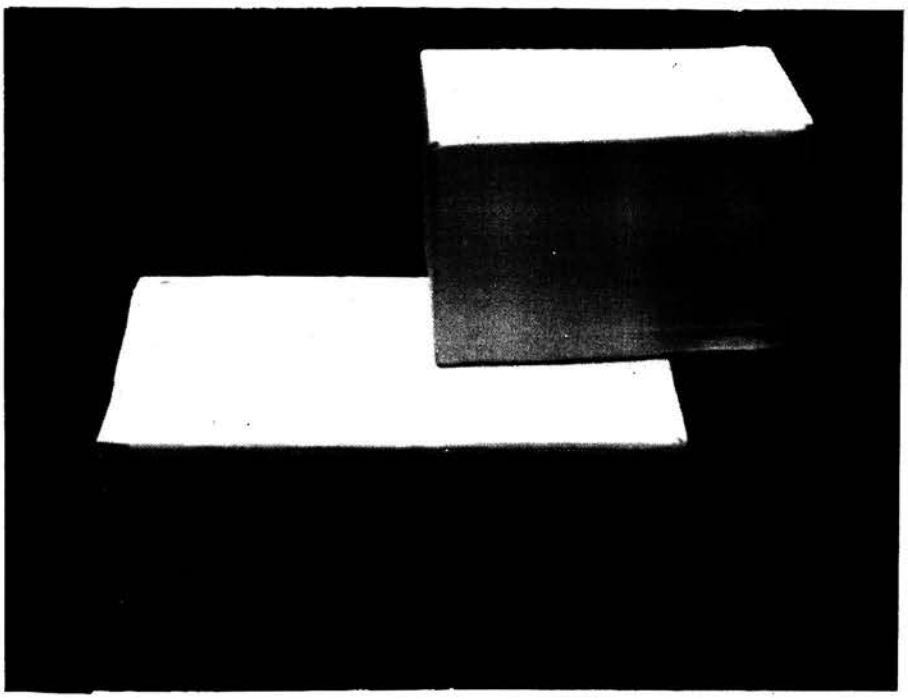
7.2 RECOGNITION

A number of examples of scenes that were recognized will be presented, and some of the problems that arose and their solution will be shown. The sorts of scenes that could be handled by the recognizer were restricted by the ranging device. The physical extent of the light plane was limited. Objects to be viewed by the ranger had to fit into a rectangular volume about 25 centimetres high by about 20 centimetres broad, with length a couple of metres. This meant that the scenes could not be too complicated because objects had to be strung out in a line rather than placed at random.

A further restriction on the kinds of scenes that could be handled by the ranger resulted from the surface-finding algorithm. In the algorithm, all surfaces that lie in a particular plane, however separated in space, are merged into one surface. This is useful, for example, if two parts of the same surface are separated by some intervening, occluding surface. When the surfaces are not related it can give rise to problems because distinct surfaces will be merged. As a result, care must be taken in placing the objects to see that distinct surfaces lie in different planes.

An example that shows most of the features of the recognition system is that of Figure 7.4. This figure shows the small box (SMALLBX) resting on top of the bigbox (BIGBOX). The surfaces that comprise BIGBOX are labelled 1 and 2 in the picture of the range map extracted from the scene. Those that belong to SMALLBX are labelled 3 and 4.

The first stage of the recognition procedure is to scan the picture and produce a range map. The surfaces produced from the range map are described in the same way as for modelling.



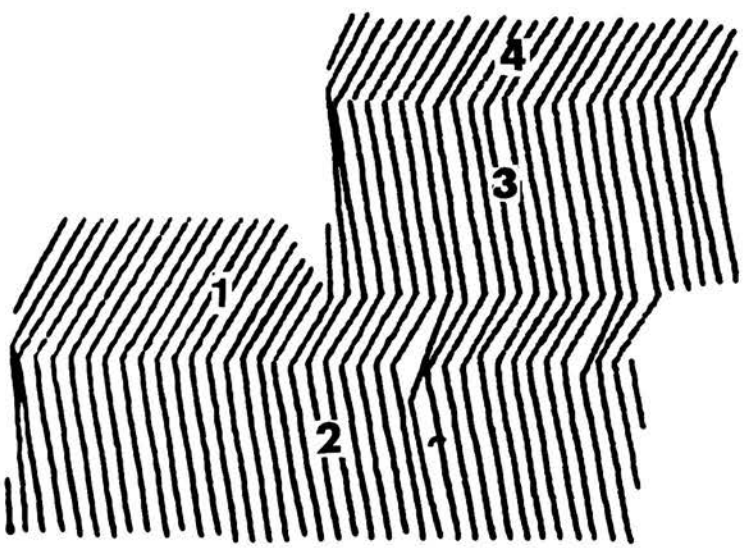


Figure 7.4

Descriptions of each surface are produced in terms of the surface type and dimensions. In this example, the dimensions are as follows:

```
RECOG([SCENE1]);
SURFACE  S1
 0.2624   0.1435
SURFACE  S2
 0.312    0.0848
SURFACE  S3
 0.1609   0.1016
SURFACE  S4
 0.1763   0.0862
```

The rest of the recognition process involves building the scene graph. This is accomplished in two stages. First, a graph is constructed that is larger than the final scene graph. This graph is then pruned to leave the graph that actually represents the objects in the scene. The first part of the process, building the graph, is guided by the graph of models. Each node in the scene graph corresponds to some node in the graph of models, although several nodes in the same graph may correspond to the same node in the graph of models. The models in the graph of models are instantiated to describe parts of the scene, forming a conventional relational structure with one part of the model for each part of the object. This was referred to earlier as the intermediate structure.

The process starts by matching the descriptions of the surfaces in the scene with the primitives in the graph of models. A surface matches with a primitive if it is of the same type and its dimensions are similar. A degree of confidence in the match, based on how similar the dimensions are, is associated with each primitive/surface pair.

In the example, surface S1 was found to be occluded. It was therefore matched with all surfaces larger than itself, with reduced confidence. The occlusion was detected because there was a sudden

change in the length of the line segments passing over it at the point at which surface S3 occluded surface S1. Thus, the matches that were found are as follows:

```
SURFACE  S1 MATCHED WITH PRIMITIVE  P2 SURFACE IS OCCLUDED
SURFACE  S2 MATCHED WITH PRIMITIVE  P4 WITH CONFIDENCE  3
SURFACE  S3 MATCHED WITH PRIMITIVE  P7 WITH CONFIDENCE  5
SURFACE  S4 MATCHED WITH PRIMITIVE  P7 WITH CONFIDENCE  5
```

Each surface is paired with the primitives with which it matched. The matches allow surfaces to index the graph of models. These primitive/surface pairs form the nodes in the scene graph. Describing a scene involves using the indices into the graph of models. The information to be extracted concerns the possible interpretations of each primitive/surface pair, and the relations between the pairs. Each pair will be assigned to one or more subgraphs of the scene graph, and each assignment will need a reason. The subgraphs in the scene graph correspond to interpretations for the surface of the pair. Each subgraph is associated with a model in the graph of models. The reasons for assignment are obtained from the graph of models. They consist of relations expected to hold between surfaces that are instances of primitives from the model.

The first step in finding an interpretation is to choose a pair to assign to the scene graph. This is done by first choosing a model that is indexed by some of the unassigned pairs, and then choosing to assign one of the pairs that index the model. In the example, the program chooses the model SMALLBX to work on, and decides to assign the pair P7/S3. Because this is the first assignment, there is no way of deciding which of the models indexed by P7 is the correct interpretation, so the pair is assigned to subgraphs associated with both of them. Even though SMALLBX suggested the assignment of P7/S3,

it has no special claim to the pair. No reasons can be given at this stage for the assignments. Notice that both possible interpretations for S3 have been found simultaneously.

```
CHOOSE SMALLBX TO WORK ON
CHOOSE PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3 TO WORK ON
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
ADDING BIGBOX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3 IN BIGBOX
  WITH ZERO CONFIDENCE
ADDING SMALLBX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3 IN SMALLBX
  WITH ZERO CONFIDENCE
```

The Set of Support is a set of the models that have been instantiated. The program tries to add pairs to subgraphs for the models in the Set of Support, rather than instantiating new models. The Set of Support contains those models that can suggest the next pair to assign.

When the program chooses the next pair to assign, it tries to choose one that will be compatible with one of its already-existing subgraphs. That is, it looks for pairs that might have the same interpretation as previously-existing pairs. First, it chooses the interpretation it would like to confirm, and then it chooses a pair that may possibly confirm the interpretation. With only two interpretations, neither of which has reasons, the choice at this stage is arbitrary. The program chooses BIGBOX to work on, and decides to assign P7/S4.

Once again, P7 indexes both SMALLBX and BIGBOX, but now it is not possible simply to assign the new pair to a subgraph for each of the models. There are already pairs assigned to subgraphs for the models, and to confirm the interpretation of those pairs, the program needs to add the new pair to an already-existing subgraph. In order to be able to do this, it has to prove that the pairs are compatible.

The program examines the graph of models and looks for relations that involve the new pair and previously-assigned pairs. It applies a relation if the success of that relation can confirm the compatibility of the new pair with a pair that has already been assigned.

In this case, two relation schemata are found with the right argument types. These are <SAMENEXT P7 P7 1> and <ADJACENT P7 P7 1>. The relation SAMENEXT is true if two surfaces are adjacent and are described by the same primitive. It gives a measure of the symmetry of an object.

The program applies both relations schemata, and both confirm the compatibility of the new pair P7/S4 with P7/S3 in the subgraph for SMALLBX. No relations succeed for BIGBOX, so P7/S4 is assigned only to the subgraph for SMALLBX.

```
CHOOSE BIGBOX TO WORK ON
CHOOSE PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S4 TO WORK ON
APPLYING RELATION SAMENEXT WITH ARGUMENTS [P7 P7] AND SURFACES [S4 S3]
RELATION SUCCEEDS
ACTIVATING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S4 IN SMALLBX
  AND GIVING IT CONFIDENCE SAMENEXT
INCREASING CONFIDENCE OF PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3
    IN SMALLBX BY SAMENEXT
APPLYING RELATION ADJACENT WITH ARGUMENTS [P7 P7] AND SURFACES [S4 S3]
RELATION SUCCEEDS
INCREASING CONFIDENCE OF PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S4
    IN SMALLBX BY ADJACENT
INCREASING CONFIDENCE OF PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3
    IN SMALLBX BY ADJACENT
```

An arc connects the two nodes in the subgraph for SMALLBX. It is labelled with the relation schemata that succeeded and confirmed the interpretation. The schemata are instantiated, and are of the form <SAMENEXT S3 S4 1> and <ADJACENT S3 S4 1>.

The program now assigns another pair. It tries to find a pair to add to the subgraph for SMALLBX, because this is the

interpretation in which it has greatest confidence. There are, however, no more unassigned pairs that index SMALLBX, so the subgraph for SMALLBX can no longer have pairs assigned to it. The program therefore removes SMALLBX from the Set of Support to prevent it trying to assign to SMALLBX later.

```
CHOOSE SMALLBX TO WORK ON
SMALLBX SATISFIED - REMOVE FROM SET OF SUPPORT
```

The only other interpretation that exists at this stage is BIGBOX. The program thus looks for a pair to assign to the subgraph for BIGBOX. It chooses P4/S2.

When relation schemata are sought to justify adding P4/S2 to the subgraph for BIGBOX (which already has P7/S3 assigned to it), it is discovered that no arcs in the graph of models join P4 to P7. There are no relation schemata that have both primitives as arguments (see Figure 7.3). This means that the two pairs cannot be compatible, and P4/S2 must be assigned to a different subgraph. P4 indexes three models, BIGBOX, MEDBOX, and CAR. P4/S2 will be assigned to subgraphs for each of these models, a second subgraph for BIGBOX, called BIGBOX1, being set up.

```
CHOOSE BIGBOX TO WORK ON
CHOOSE PRIMITIVE  P4 ASSOCIATED WITH SURFACE  S2 TO WORK ON
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
FORMING ANOTHER INSTANCE OF BIGBOX
ADDING BIGBOX1 TO SET OF SUPPORT
ADDING PRIMITIVE  P4 ASSOCIATED WITH SURFACE  S2 TO BIGBOX1
  WITH ZERO CONFIDENCE
ADDING MEDBOX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P4 ASSOCIATED WITH SURFACE  S2 IN MEDBOX
  WITH ZERO CONFIDENCE
ADDING CAR TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P4 ASSOCIATED WITH SURFACE  S2 IN CAR
  WITH ZERO CONFIDENCE
```

Finally, the program assigns P2/S1. At this stage there are four possible subgraphs to which P2/S1 could be assigned - MEDBOX, CAR,

BIGBOX, and BIGBOX1. The program looks for relations between surfaces of type P2 and those of type P4 and P7. It finds one ADJACENT relation schema for each of P4 and P7, and both of these succeed when they are applied to their paired surfaces.

The fact that S1 is ADJACENT to S2 is evidence that P2/S1 should be added to the subgraph for BIGBOX1. Similarly S1 is ADJACENT to S3, and this causes P2/S1 to be added to the subgraph for BIGBOX. No changes are made to the subgraphs for CAR or MEDBOX.

```
CHOOSE BIGBOX TO WORK ON
CHOOSE PRIMITIVE  P2 ASSOCIATED WITH SURFACE  S1 TO WORK ON
APPLYING RELATION ADJACENT WITH ARGUMENTS [P2 P4] AND SURFACES [S1 S2]
RELATION SUCCEEDS
ACTIVATING PRIMITIVE  P2 ASSOCIATED WITH SURFACE  S1 IN BIGBOX1
INCREASING CONFIDENCE OF PRIMITIVE  P4 ASSOCIATED WITH SURFACE  S2
    IN BIGBOX1 BY ADJACENT
APPLYING RELATION ADJACENT WITH ARGUMENTS [P2 P7] AND SURFACES [S1 S3]
RELATION SUCCEEDS
INCREASING CONFIDENCE OF PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3
    IN BIGBOX BY ADJACENT
ACTIVATING PRIMITIVE  P2 ASSOCIATED WITH SURFACE  S1 IN BIGBOX
```

Notice that S1 and S3 are actually from different objects (Figure 7.4) and are only coincidentally related. This relationship arises because the long side of SMALLBX (S1) is described in exactly the same way (by the same primitive) as the short side of BIGBOX (not visible). When BIGBOX was modelled, the ADJACENT schema was constructed to describe the relationship between the biggest and smallest sides of BIGBOX. This relation schema was discovered and applied during recognition.

The second stage of the process is the constraint analysis phase. The scene graph has been completely built, but several pairs have been assigned to more than one subgraph and so have more than one interpretation. The ambiguous interpretations are examined and those that are best substantiated are retained.

The program finds that surface S4 has a unique interpretation, but that surface S3 has two. Surface S3 could either be part of SMALLBX or of BIGBOX. To decide between the two interpretations, the system examines the reasons for assignment. It observes that two relations (ADJACENT and SAMENEXT) succeeded, binding surface S3 to surface S4 in SMALLBX. Only one relation (ADJACENT) succeeded, binding surface S3 to surface S1 in BIGBOX. The program decides that this difference is sufficient reason to discard the interpretation of surface S3 as part of BIGBOX. It therefore deletes the node P7/S3 from the subgraph for BIGBOX. Deleting the node means that the arc connecting P7/S3 to P2/S1 in the subgraph must be deleted also, and the relation on the arc must be discarded. This, in turn, means that P2/S1 no longer has any reasons for being interpreted as part of BIGBOX.

```
BODIES IN WHICH  P7 ASSOCIATED WITH SURFACE  S4 ARE: [SMALLBX]
BODIES IN WHICH  P7 ASSOCIATED WITH SURFACE  S3 ARE: [SMALLBX BIGBOX]
DEACTIVATING  P7 ASSOCIATED WITH SURFACE  S3 IN BIGBOX
FILTERING - REMOVING RELATION ADJACENT IN BIGBOX
  RELATING PRIMITIVES [P2 P7] ASSOCIATED WITH SURFACES [S1 S3]
```

Surface S2 was assigned to three subgraphs, those for CAR, MEDBOX, and BIGBOX1. The surface was found to be compatible with surface S1 in BIGBOX1, but there were no reasons for its assignment to the other subgraphs. As a result, the subgraphs for CAR and MEDBOX are removed from the scene graph.

```
BODIES IN WHICH  P4 ASSOCIATED WITH SURFACE  S2 ARE: [CAR MEDBOX
 BIGBOX1]
DEACTIVATING  P4 ASSOCIATED WITH SURFACE  S2 IN MEDBOX
REMOVING MEDBOX FROM ACTIVE BODIES
DEACTIVATING  P4 ASSOCIATED WITH SURFACE  S2 IN CAR
REMOVING CAR FROM ACTIVE BODIES
```

Finally, surface 1 has interpretations as part of BIGBOX and of BIGBOX1. Initially, both interpretations were equally well

substantiated (by ADJACENT relations). However, the filtering step that decided on the interpretation of surface S3 also eroded the reasons for the interpretation of surface S1. Thus, surface S1 now has no reasons for being part of BIGBOX, and is interpreted as part of BIGBOX1.

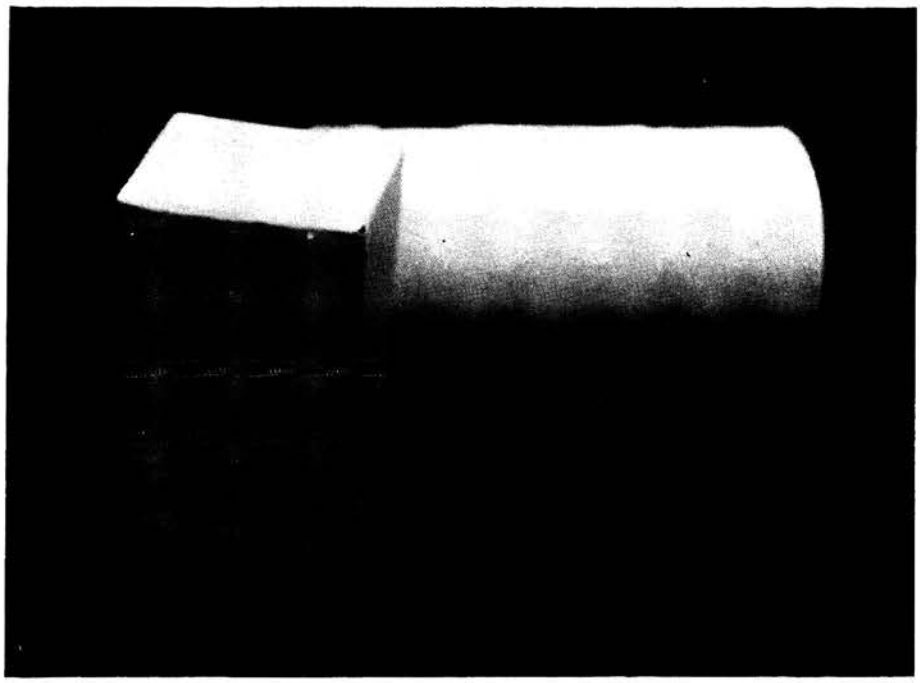
```
BODIES IN WHICH  P2 ASSOCIATED WITH SURFACE  S1 ARE: [BIGBOX1 BIGBOX]
DEACTIVATING  P2 ASSOCIATED WITH SURFACE  S1 IN BIGBOX
REMOVING BIGBOX FROM ACTIVE BODIES
```

The final interpretation is

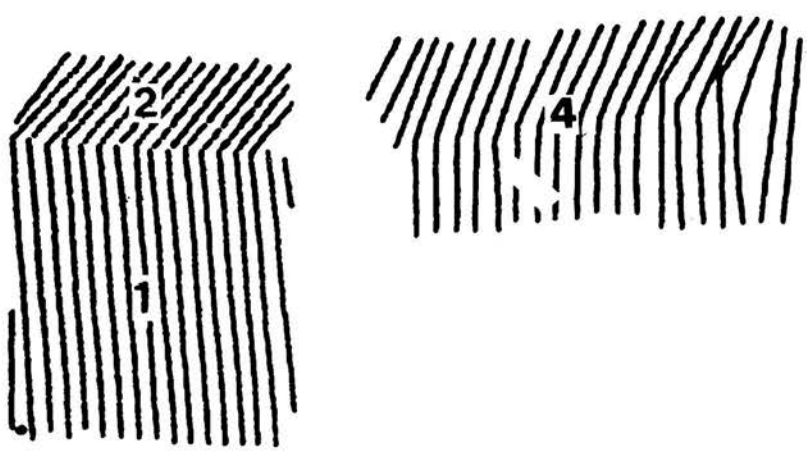
```
SMALLBX
 S4
ADJACENT  S4  S3
SAMENEXT  S4  S3
 S3
ADJACENT  S4  S3
SAMENEXT  S4  S3

BIGBOX1
 S1
ADJACENT  S1  S2
 S2
ADJACENT  S1  S2
```

Figure 7.5a depicts the small box in front of the cylinder, while Figure 7.5b shows the result of scanning the scene and segmenting each stripe into straight line segments. Note how the cylinder has been segmented into planar facets. The large gap between the segments describing the box and those of the cylinder is caused by joint shadowing from the camera and the light source.

Figure 7.6 shows a trace of the action of the recognizer in interpreting the scene. The output consists of the name of each object recognized, the set of surfaces that make up the object, and the relations that served as evidence for assigning the surfaces to the object.



(a)



(b)

Figure 7.5

```
RECOG([SCENE2]);

Read in the surface data and process it to find the dimensions
of each surface.

SURFACE  S1
 0.1367   0.0938
SURFACE  S2
 0.0987   0.0881
SURFACE  S4
 0.1169   0.0508

Match the surfaces in the scene with the primitives in the
graph of models.

SURFACE  S1 MATCHED WITH PRIMITIVE  P7 WITH CONFIDENCE  5
SURFACE  S2 MATCHED WITH PRIMITIVE  P5 WITH CONFIDENCE  5
SURFACE  S4 MATCHED WITH PRIMITIVE  P6 WITH CONFIDENCE  2

Assign the primitive/surface pairs to model instances.

PRIMITIVE  P6 ASSOCIATED WITH SURFACE  S4 HAS CYL AS ONLY CANDIDATE
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
ADDING CYL TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P6 ASSOCIATED WITH SURFACE  S4 IN CYL
  WITH ZERO CONFIDENCE
CHOOSE SMALLBX TO WORK ON
CHOOSE PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S1 TO WORK ON
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
ADDING BIGBOX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S1 IN BIGBOX
  WITH ZERO CONFIDENCE
ADDING SMALLBX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S1 IN SMALLBX
  WITH ZERO CONFIDENCE
CHOOSE SMALLBX TO WORK ON
CHOOSE PRIMITIVE  P5 ASSOCIATED WITH SURFACE  S2 TO WORK ON

When a subgraph already has a pair assigned to it, relation tests
must be applied before another pair can be assigned.

APPLYING RELATION ADJACENT WITH ARGUMENTS [P7 P5] AND SURFACES [S1
 S2]
RELATION SUCCEEDS
ACTIVATING PRIMITIVE  P5 ASSOCIATED WITH SURFACE  S2 IN SMALLBX
INCREASING CONFIDENCE OF PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S1
    IN SMALLBX BY ADJACENT

All the relations have been tested. Now the interpretations
for each pair are examined.
```

Figure 7.6

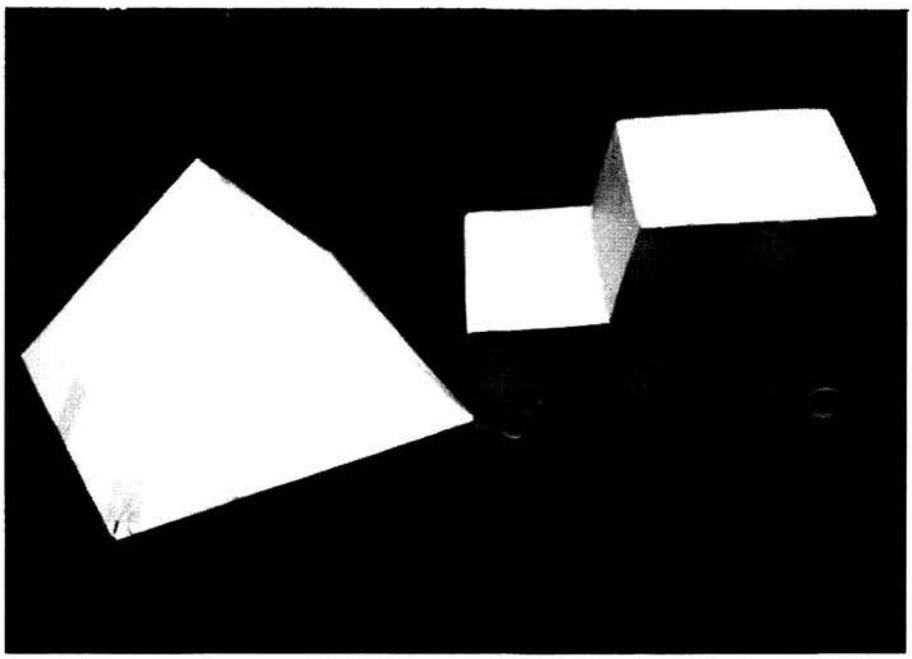
```
BODIES IN WHICH  5 ASSOCIATED WITH SURFACE  2 ARE: [SMALLBX]
BODIES IN WHICH  7 ASSOCIATED WITH SURFACE  1 ARE: [SMALLBX BIGBOX]

BIGBOX is discarded as an interpretation because of the successful
relation tests in SMALLBX.

DEACTIVATING  7 ASSOCIATED WITH SURFACE  1 IN BIGBOX
REMOVING BIGBOX FROM ACTIVE BODIES
BODIES IN WHICH  6 ASSOCIATED WITH SURFACE  4 ARE: [CYL]

The final interpretation is:

SMALLBX
 S2
ADJACENT  S2  S1
 S1
ADJACENT  S2  S1

CYL
 S4

DO YOU WANT TO TRY HARDER? (Y OR N):N
```

Figure 7.6(cont'd)

The next example shows the use of the rematching facility for trying to improve the result after recognition (section 6.4.1). In Figure 7.7a we see the pyramid to the left of the car. The segments produced by scanning this scene are shown in Figure 7.7b. The roof of the car (surface S5) was incorrectly measured, and matched with primitive P7 which indexes only BIGBOX and SMALLBX (see Figure 7.3). Figure 7.8 shows how the scene was recognized.

Surface S5 did not take part in any relation tests, and was assigned to instances of BIGBOX and SMALLBX with zero confidence. The rest of the scene was correctly analyzed. When the chance of trying harder was presented, only surface S5 was eligible for rematching (the others all had reasons for assignment). On rematching, with relaxed criteria, the correct primitive was included among the matches, and so surface S5 was added to the instance of CAR, giving the final result.

Another example shows an aspect of the robustness of the recognition. The scene is that of Figure 7.9 and the thing to note is that in Figure 7.10 both SURFACE S4 and SURFACE S5 were clustered anomalously, so that the input routine found two views. In the case of SURFACE S5 this made little difference, since the correct primitive match was found. SURFACE S4, however, was badly affected, and matched with five different primitives. Therefore, a lot of assignments were made and a lot of relations tested. The constraint analyzer, however, was able to discard most of the interpretations, giving the final result.



(a)

(b)

Figure 7.7

```
RECOG([SCENE3]);

Find the dimensions of the surfaces.

SURFACE  S1
 0.1609   0.142
SURFACE  S2
 0.1391   0.12
SURFACE  S3
 0.2347   0.1116
SURFACE  S5
 0.1237   0.1075
SURFACE  S6
 0.1959   0.1512

Match the surfaces with primitives in the graph of models.
Notice that surface S5 finds the wrong match.

SURFACE  S1 MATCHED WITH PRIMITIVE  P1 WITH CONFIDENCE  5
SURFACE  S2 MATCHED WITH PRIMITIVE  P7 WITH CONFIDENCE  5
SURFACE  S2 MATCHED WITH PRIMITIVE  P1 WITH CONFIDENCE  5
SURFACE  S3 MATCHED WITH PRIMITIVE  P4 WITH CONFIDENCE  5
SURFACE  S5 MATCHED WITH PRIMITIVE  P7 WITH CONFIDENCE  5
SURFACE  S6 MATCHED WITH PRIMITIVE  S2 SURFACE IS OCCLUDED

Start assigning pairs to subgraphs.

CHOOSE PYRAMID TO WORK ON
CHOOSE PRIMITIVE  P1 ASSOCIATED WITH SURFACE  S1 TO WORK ON
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
ADDING CAR TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P1 ASSOCIATED WITH SURFACE  S1 IN CAR
  WITH ZERO CONFIDENCE
ADDING PYRAMID TO SET OF SUPPORT
ACTIVATING PRIMITIVE  Pi ASSOCIATED WITH SURFACE  S1 IN PYRAMID
  WITH ZERO CONFIDENCE
CHOOSE CAR TO WORK ON
```

.

.

.

The usual sequence of choosing pairs and assigning them to subgraphs on the basis of relational tests occurs here.

.

.

.

Figure 7.8

Surface S5 is finally chosen for assignment.

```
CHOOSE SMALLBX TO WORK ON
CHOOSE PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S5 TO WORK ON
APPLYING RELATION SAMENEXT WITH ARGUMENTS [P7 P7] AND SURFACES [S5
 S2]
RELATION FAILS
APPLYING RELATION ADJACENT WITH ARGUMENTS [P7 P7] AND SURFACES [S5
 S2]
RELATION FAILS
```

No relations succeed because the wrong match was found for S5. The pair is assigned to new subgraphs for the models it indexes. No reasons are given for the assignments.

```
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
FORMING ANOTHER INSTANCE OF BIGBOX
ADDING BIGBOX1 TO SET OF SUPPORT
ADDING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S5 TO BIGBOX1
  WITH ZERO CONFIDENCE
FORMING ANOTHER INSTANCE OF SMALLBX
ADDING SMALLBX1 TO SET OF SUPPORT
ADDING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S5 TO SMALLBX1
  WITH ZERO CONFIDENCE
```

.

.

.

The rest of the surfaces are assigned in the usual way.

.

.

.

The constraint analysis stage begins. Neither interpretation of surface S5 has any reasons and both are retained as ambiguous interpretations.

```
BODIES IN WHICH  P7 ASSOCIATED WITH SURFACE  S5 ARE: [SMALLBX1
 BIGBOX1]
BODIES IN WHICH  P4 ASSOCIATED WITH SURFACE  S3 ARE: [CAR1]
BODIES IN WHICH  P7 ASSOCIATED WITH SURFACE  S2 ARE: [PYRAMID1
 PYRAMID
CAR1 CAR SMALLBX BIGBOX]
DEACTIVATING  P7 ASSOCIATED WITH SURFACE  S2 IN BIGBOX
REMOVING BIGBOX FROM ACTIVE BODIES
DEACTIVATING  P7 ASSOCIATED WITH SURFACE  S2 IN SMALLBX
REMOVING SMALLBX FROM ACTIVE BODIES
```

Figure 7.8(cont'd)

As usual, those interpretations with insufficient reasons are deleted, giving this result at the end:

```
SMALLBX1
 S5

BIGBOX1
 S5

CAR1
 S2
ADJACENT  S3  S2
 S3
ADJACENT  S3  S2

PYRAMID
 S6
RELDIST  S6  S1
 S1
RELDIST  S6  S1

DO YOU WANT TO TRY HARDER? (Y OR N):Y
```

This time, when the chance of trying harder is presented, it is taken. Surface S5 is rematched with the primitives in the graph of models, with relaxed criteria. The assumption made is that the surface is occluded.

```
SURFACE  S5 MATCHED WITH PRIMITIVE  P1 SURFACE IS OCCLUDED
SURFACE  S5 MATCHED WITH PRIMITIVE  P2 SURFACE IS OCCLUDED
```

The recognition algorithm restarts, and assignments are made in the usual way.

```
CHOOSE CAR1 TO WORK ON
CHOOSE PRIMITIVE  P1 ASSOCIATED WITH SURFACE  S5 TO WORK ON
APPLYING RELATION ADJACENT WITH ARGUMENTS [P1 P4] AND SURFACES [S5
 S3]
RELATION SUCCEEDS
ACTIVATING PRIMITIVE  P1 ASSOCIATED WITH SURFACE  S5 IN CAR1
  AND GIVING IT CONFIDENCE ADJACENT
INCREASING CONFIDENCE OF PRIMITIVE  P4 ASSOCIATED WITH SURFACE  S3
    IN CAR1 BY ADJACENT
APPLYING RELATION RELDIST WITH ARGUMENTS [P1 P2] AND SURFACES [S5 S6]
RELATION FAILS
```

Figure 7.8(cont'd)

```
All relation tests relevant to the new pairs are made, and
all assignments to subgraphs are made.
Finally, the constraint analysis phase is entered again.

BODIES IN WHICH  P2 ASSOCIATED WITH SURFACE  S5 ARE: [PYRAMID CAR1
PYRAMID1 BIGBOX]
DEACTIVATING  P2 ASSOCIATED WITH SURFACE  S5 IN BIGBOX
REMOVING BIGBOX FROM ACTIVE BODIES
DEACTIVATING  P2 ASSOCIATED WITH SURFACE  S5 IN PYRAMID1
REMOVING PYRAMID1 FROM ACTIVE BODIES
DEACTIVATING  P1 ASSOCIATED WITH SURFACE  S5 IN PYRAMID
BODIES IN WHICH  P1 ASSOCIATED WITH SURFACE  S5 ARE: [PYRAMID CAR1]
DEACTIVATING  P2 ASSOCIATED WITH SURFACE  S5 IN PYRAMID

The final interpretation for surface S5 is the correct one:

CAR1
 S5
ADJACENT  S5  S3
 S2
ADJACENT  S3  S2
 S3
ADJACENT  S5  S3
ADJACENT  S3  S2

DO YOU WANT TO TRY HARDER? (Y OR N):N
```

Figure 7.8(cont'd)

Even though the final set is ambiguous, all objects in the scene have been correctly identified, and the correct surfaces have been associated with each object. The only thing the algorithm has been unable to accomplish is to decide on the exact identity of surface 4. Even that has been reduced to two alternatives.

In the following chapter, examples will be given using a larger database of models, but in a domain where the difficulties of the early analysis are removed. The two sets of results from different domains serve to illustrate the power and flexibility of the modelling and recognition techniques.

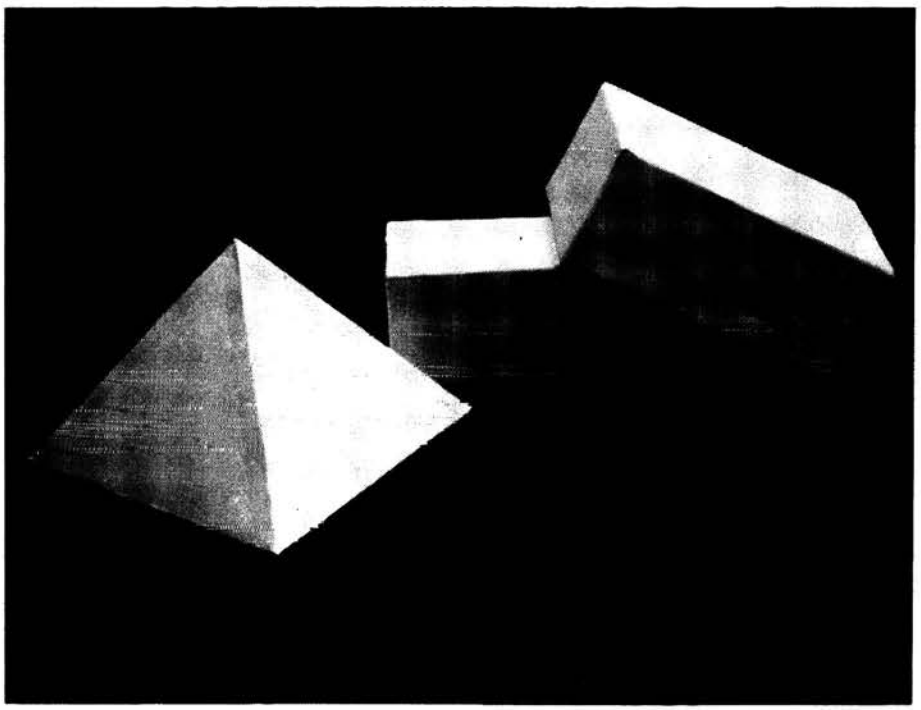


Figure 7.9

```
RECOG([SCENE4]);
SURFACE  S1
 0.2286   0.177
SURFACE  S2
 0.1131   0.093
SURFACE  S3
 0.1463   0.091
```

Surface S4 and surface S5 are misinterpreted. The program assumes that there are two view of each surface, and so obtains incorrect dimensions for them.

```
SURFACE  S4
NEW VIEW  S4
 0.119   0.0717
SURFACE  S5
NEW VIEW  S5
 0.1762   0.1471
SURFACE  S6
 0.2067   0.1556
```

When surfaces are matched with primitives in the graph of models, surface S4 matches with five primitives, introducing a lot of ambiguity.

```
SURFACE  S1 MATCHED WITH PRIMITIVE  P2 SURFACE IS OCCLUDED
SURFACE  S2 MATCHED WITH PRIMITIVE  P5 WITH CONFIDENCE  5
SURFACE  S3 MATCHED WITH PRIMITIVE  P7 WITH CONFIDENCE  5
SURFACE  S4 MATCHED WITH PRIMITIVE  P1 SURFACE IS OCCLUDED
SURFACE  S4 MATCHED WITH PRIMITIVE  P2 SURFACE IS OCCLUDED
SURFACE  S4 MATCHED WITH PRIMITIVE  P3 SURFACE IS OCCLUDED
SURFACE  S4 MATCHED WITH PRIMITIVE  P4 SURFACE IS OCCLUDED
SURFACE  S4 MATCHED WITH PRIMITIVE  P7 SURFACE IS OCCLUDED
SURFACE  S5 MATCHED WITH PRIMITIVE  P2 SURFACE IS OCCLUDED
SURFACE  S6 MATCHED WITH PRIMITIVE  P2 SURFACE IS OCCLUDED
CHOOSE MEDBOX TO WORK ON
CHOOSE PRIMITIVE  P5 ASSOCIATED WITH SURFACE  S2 TO WORK ON
ASSIGNING PRIMITIVE TO ALL CANDIDATES WITH ZERO CONFIDENCE
ADDING SMALLBX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P5 ASSOCIATED WITH SURFACE  S2 IN SMALLBX
  WITH ZERO CONFIDENCE
ADDING MEDBOX TO SET OF SUPPORT
ACTIVATING PRIMITIVE  P5 ASSOCIATED WITH SURFACE  S2 IN MEDBOX
  WITH ZERO CONFIDENCE
CHOOSE SMALLBX TO WORK ON
CHOOSE PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3 TO WORK ON
APPLYING RELATION ADJACENT WITH ARGUMENTS [P7 P5] AND SURFACES [S3
 S2]
RELATION SUCCEEDS
INCREASING CONFIDENCE OF PRIMITIVE  P5 ASSOCIATED WITH SURFACE  S2
    IN SMALLBX BY ADJACENT
ACTIVATING PRIMITIVE  P7 ASSOCIATED WITH SURFACE  S3 IN SMALLBX
```

Figure 7.10

A large number of assignments and relational tests are made. Surface S4 receives 12 interpretations, all but two having zero confidence.

```
BODIES IN WHICH  P7 ASSOCIATED WITH SURFACE  S3 ARE: [SMALLBX]
BODIES IN WHICH  P5 ASSOCIATED WITH SURFACE  S2 ARE: [MEDBOX SMALLBX]
DEACTIVATING  P5 ASSOCIATED WITH SURFACE  S2 IN MEDBOX
REMOVING MEDBOX FROM ACTIVE BODIES
BODIES IN WHICH  P2 ASSOCIATED WITH SURFACE  S6 ARE: [PYRAMID1]
BODIES IN WHICH  P2 ASSOCIATED WITH SURFACE  S5 ARE: [BIGBOX1 BIGBOX]
BODIES IN WHICH  P7 ASSOCIATED WITH SURFACE  S4 ARE:
[PYRAMID CAR1 PYRAMID1 PYRAMID2 BIGBOX1 BIGBOX2 BIGBOX3
BIGBOX CAR2 CAR MEDBOX1 BIGBOX1 SMALLBX1 BIGBOX]
```

Figure 7.10(cont'd)

All interpretations for surface S4 except BIGBOX and BIGBOX1 have no reasons, and are deleted, giving the final result

```
SMALLBX
 S2
ADJACENT  S3  S2
 S3
ADJACENT  S3  S2

BIGBOX1
 S5
ADJACENT  S5  S4
 S4
ADJACENT  S5  S4

BIGBOX
 S4
ADJACENT  S5  S4
 S5
ADJACENT  S5  S4

PYRAMID1
 S6
RELDIST  S6  S1
RELDIST  S6  S1
 S1
RELDIST  S6  S1
RELDIST  S6  S1

DO YOU WANT TO TRY HARDER? (Y OR N):N
```

Figure 7.10

8.0 SPELLING CORRECTION

In order to demonstrate the versatility, generality and limitations of the representation and recognition methods described in this thesis, it was decided to apply the system to another domain - that of spelling correction.

There are a number of parallels between the applications: Words are self-contained in much the same way as solid objects are, and are amenable to description in similar terms. Some of the variations from the correct spelling that can occur have their analogues in the visual domain. For instance, missing letters correspond to missing surfaces.

There are, however, significant differences. The major of these is that the spelling problem is essentially one-dimensional. Words are sequential strings of letters, with no significant relationships other than order. It is often easy to find the end of one word and the beginning of the next, by making use of the convention that words are separated by spaces. Having done this, it is possible to deal with each word separately. To some extent, too, extra letters can be found to be extraneous and ignored (because letters of one word cannot be interspersed with those of another). This is not the case in vision where any single surface might be the only visible part of an object, and might occur in almost any relationship with surfaces

of other objects.

8.1 THE PROBLEM

It was desired to construct models of correct words, and then use the models to correct the spelling of instances of the words. It was expected that most kinds of typographic errors could be handled, most notably:

1. Missing letters.
2. Extra letters.
3. Transposed letters.
4. Incorrect letters.
5. Words run together without spaces.
6. Ambiguous letters. This presupposes the existence of a preprocessor that could deliver letters or words from a visual inspection. In many cases such a system would not be able to distinguish letters unambiguously (e.g. a character might be seen as either an E or an F). The recognition system would be expected to disambiguate such letters by means of their context (Bornat(1976), Brady and Wielinga(1979)).

In addition to these corrections, the system was found to have other abilities. Presenting a sequence of letters that is not a complete word will result in the system returning all words that contain that string as a substring. The system is also proficient at discovering anagrams. Damerau (1964) noted that 80% of spelling errors fall into the first four categories listed above. This system, with its extra capabilities, can be expected to correct

almost all common errors that do not require an understanding of the context.

8.2 WORD MODELS

Words are modelled in an analogous way to objects. In the case of words, letters are taken as primitives, and various ordering and distinguishing relations are defined between them. An example of a model for a word is shown in Figure 8.1.

Several relations have been used in describing words. Some of these are:

STARTWRD and ENDWORD: These are predicates. They are true if the letter is the first (or last) of a word. Predicates are represented as binary relations with both arguments identical.

BEFORE: This relation is true if its first argument occurs earlier in the sequence of letters than its second.

PRECEDES: This relation is true only if the first argument occurs immediately before the second in the string of characters.

A corresponding pair of functions, AFTER and FOLLOWS, have also been defined. They are equivalent to BEFORE and PRECEDES respectively.

At first glance it might appear redundant to have both BEFORE and PRECEDES as relations describing words. There are, however, cases in which each serves a role that the other is unable to fulfil. If PRECEDES only were defined, and a word had an extra letter in it, then a chain of assignments to an instance of the model for that word would be broken at that letter. No relations would succeed. The result would be to produce two copies of the word, instead of one.

```
PRIMITIVES:


S  P1
BODYLIST 'SEE!
EXTENT  0.0     0.0

E  P2
BODYLIST 'SEE!
EXTENT  0.0     0.0


WORD:


'SEE!
DESCRIPTION
E  P2
APPEARS IN RELATIONS: PRECEDES R6 BEFORE R5 PRECEDES R4 BEFORE R3
 ENDWORD R2
S  P1
APPEARS IN RELATIONS: PRECEDES R4 BEFORE R3 STARTWRD R1


RELATIONS:


PRECEDES  R6  ARGUMENTS  P2   P2
EXPECTED VALUE  1.0
BODYLIST 'SEE!

BEFORE  R3  ARGUMENTS  P1   P2
EXPECTED VALUE  1.0
BODYLIST 'SEE!

STARTWRD  R1  ARGUMENTS  P1   P1
EXPECTED VALUE  1.0
BODYLIST 'SEE!

ENDWORD  R2  ARGUMENTS  P2   P2
EXPECTED VALUE  1.0
BODYLIST 'SEE!

PRECEDES  R4  ARGUMENTS  P1   P2
EXPECTED VALUE  1.0
BODYLIST 'SEE!

BEFORE  R5  ARGUMENTS  P2   P2
EXPECTED VALUE  1.0
BODYLIST 'SEE!
```

Figure 8.1

Using BEFORE on its own also causes problems. Two words, such as ASSES and ASSESS, might have the same letters in different positions, but all relations in the one might occur in the other. These words would be indistinguishable using only BEFORE. Also, because the system does not know how many letters there are in a word, it would be unable to distinguish words like ATE and TASTE if the former were presented.

STARTWRD and ENDWORD were included to illustrate the use of predicates. They may, however, be considered to give undue weight to the first and last letters when identifying a word.

8.3 IMPLEMENTATION.

The implementation was simply a matter of writing a new input routine to massage the data into a form similar to that used by the vision system. The output, also, was modified to read in a suitable manner.

Words are presented as strings of letters for learning. The string is broken into individual letters and sent to the modeller. For recognition, a word or string of words is typed in, terminated by a full stop. Each word is recognized, and any errors corrected. The output is typed in the same format as those of the input, in upper case. Formatting and punctuation may be the same as the input. Spaces, newlines, and punctuation marks can all be learned in the same way as words.

Note that both word and object models can coexist in the database. Recognition in either domain will be unaffected.

8.4 EXAMPLES

The examples will be divided into three groups. In the first an arbitrarily-chosen set of words will be used. In the second group of examples, a set of words which are very similar will be used. The third group will be taken from spelling errors made by children in the first few years of learning to write (ages 5-7). It will be seen that performance is excellent for words in the first and third groups, and satisfactory for the second sample.

The first set of words was taken from "Alice in Wonderland" by Lewis Carroll(1960). It consists of the words and punctuation of the verse:

"Speak roughly to your little boy,
And beat him when he sneezes:
He only does it to annoy,
Because he knows it teases."

Figure 8.2a shows simply the recognition of correctly-spelled words.

Figure 8.2b shows the same sentence with several misspellings of various sorts. The word "spik" for "speak" shows both missing letters and extra (or incorrect) letters, both of which are corrected. "ruffly" is similar, as is "yor", while "littel" illustrates transposition.

Figure 8.2c shows two new types of errors in addition to those already observed. The string "onlidus" consists of two (misspelled) words run together, and is correctly rendered as ONLY DOES. The second kind of error is demonstrated by the string "t[ef]sess". Here the square brackets signal uncertainty about the identity of their contents. That is, the second letter in the string is either an "e" or an "f". The system is expected to discover which (if any) of the

:Speak roughly to your little boy.
SPEAK ROUGHLY TO YOUR LITTLE BOY.

(a)

:spik ruffly to yor littel boy.
SPEAK ROUGHLY TO YOUR LITTLE BOY.

(b)

:He onlidus it to anoy becos he nos it t[ef]sess.
HE ONLY DOES IT TO ANNOY BECAUSE HE KNOWS IT TEASES.

(c)

:hee ony do it to anoii 'cause he knose it teazs.
HE ONLY DOES IT TO ANNOY BECAUSE HE KNOWS IT SNEEZES TEASES.

(d)

:spekeruffly tooo yor litel boi,
and beet him wen hee sne[efk]z:
he onli do it to anoi,
cause hee knos it teasiss.
SPEAK ROUGHLY TO YOUR LITTLE BOY,
AND BEAT HIM WHEN HE SNEEZES:
HE ONLY DOES IT TO ANNOY,
BECAUSE HE KNOWS IT TEASES.

(e)

:ezesens.
SNEEZES.

(f)

:ksape gohrlyu to yruo itltle ybo.
SPEAK ROUGHLY TO YOUR LITTLE BOY.

(g)

Figure 8.2

alternatives is correct. It does this by means of the linking mechanism that was used for uncertain matches and occlusion in the visual domain.

Figures 8.2d-e provide further examples of the system, which illustrate the various kinds of error correction of which it is capable. Note that in Figure 8.2d it was unable to decide unambiguously on the identity of "teazs", and so returned both "teases" and "sneezes" as possibilities.

Figures 8.2f-g show the recognition of various words from anagrams of their letters. Anagrams are identified largely because, in almost any rearrangement of a word, some letters will be in expected relationships. Given that this is the case, it is usually true that the most likely candidate is the correct word.

When we come to the domain of specially chosen words, the error-correction is not quite as good. The domain consists of the words

"BASTE THE BEAST FOR THE EASTER FEAST"

chosen for the great similarity of the words. "BASTE" and "BEAST" are anagrams, while four of the words have the letters E, A, S, T in them.

Figure 8.3a shows the recognition of the sentence with correct spelling. Later examples, however, show a much greater variation in correctness. Some of the errors are worth analyzing.

Figure 8.3b is correctly recognized since letters appear in the correct order, even though some are missing or added.

Figure 8.3c shows how an incorrectly spelled word can be

:Baste the beast for the Easter feast.
BASTE THE BEAST FOR THE EASTER FEAST.

(a)

:bste thet best fr th estr fest.
BASTE THE BEAST FOR THE EASTER FEAST.

(b)

:ase thebas foteh easr fest.
BASTE THE BASTE FOR THE EASTER FEAST.

(c)

:Bastethe baest fo teh este fast.
EASTER BEAST FOR THE EASTER FEAST.

(d)

:ase thbst foteh easr fst.
BASTE THE BEAST FOR THE EASTER FEAST.

(e)

:[bf]east [eb]aaster [eb]aste.
FEAST BEAST EASTER BASTE.

(f)

:as.
BASTE BEAST EASTER FEAST.

(g)

:st.
BEAST FEAST.

(h)

Figure 8.3

confused with a similar word in the database. The string "thebas" is split into THE and into BASTE instead of BEAST. This is obviously the correct answer for the recognizer, measured using the set of relations with which the words were learned (PRECEDES is successful here, adding confidence to BASTE but not to BEAST).

With a set of relations excluding PRECEDES, both BASTE and BEAST would have been ambiguously returned, but there is no useful set that would return only BEAST. To do that would require knowledge of the meaning of the alternatives, and an understanding of the sentence.

Figure 8.3d illustrates how the system can fail due to the incompleteness of the tree search during filtering. Having once decided to assign a letter to a word, the system is committed to the consequences. In this case, the order of such assignments becomes critical. When the filtering is begun, the candidates for the string "Bastethe" are THE, EASTER, FEAST, BASTE, and BEAST. Unfortunately, the word THE has a poor chance of survival, since both the T and the E are related to many of the letters in the preceding string "Baste". Because the program has no knowledge of how many times a letter occurs in a word, it happily discards these letters from THE because they appear to fit better with the other candidates. All that remains associated with THE is the H. The output routine suppresses THE as an interpretation because single letter interpretations are assumed to arise from extra letters in misspelled words.

BEAST and FEAST can be removed from consideration because they are not as well supported as BASTE and EASTER. The system works on a letter-by-letter basis, and words that are poor candidates are depleted of their letters one by one, until none remain, and the word can be removed from consideration.

In this example, the order in which the letters are examined is important in deciding the final result. For instance, the fourth letter, T, has better reasons for being in BASTE than in EASTER, while the fifth, E, has better reasons for being in EASTER. It so happens that the program chooses to look at the fifth letter first, and, by a process of attrition, ends up with EASTER as its final result. The other result would have followed a choice of the T (or of any other letter). If both alternatives had been followed, a different and correct result would have been found, since BASTE has a clear win over EASTER in all letters but the E.

Figure 8.3e shows another correct recognition of the sentence. Figure 8.3f shows three examples of interpreting ambiguous letters. In all cases the result is either the correct word or, when there are several possibilities, the set of correct words.

Figure 8.3g shows how a substring will cause the recognizer to return all words which contain that substring, although Figure 8.3h shows that when the predicates STARTWRD and ENDWORD are known, the result is the subset of words that starts with the first letter of the substring, or ends with the last letter. To be sure of finding all words in such a case, it is necessary to surround the substring with a character that does not appear in the words.

A third class of examples was taken from errors made by young schoolchildren. A selection of short contributions to the school magazine of St Margaret's School, Newington, in Edinburgh, was used. Figure 8.4 shows how in all cases the errors in spelling were corrected. The contributions were treated in two batches. The first and third were handled together, and the rest comprised the second group.

:I went to Majorca and I petAd a doncA and
it cicd the ceapr and he had a soor TumA.
I WENT TO MAJORCA AND I PETTED A DONKEY
AND IT KICKED THE KEEPER AND HE HAD A SORE TUMMY .

:My PaRty is to-day not the hol class is caming.
MY PARTY IS TODAY NOT THE WHOLE CLASS IS COMING.

:She swepd the flors she dusted the windows her
ugly sisters war avited to a dan.
SHE SWEPT THE FLOORS SHE DUSTED THE WINDOWS HER
UGLY SISTERS WERE INVITED TO A DANCE.

:hail cam dun the chimye yesterday.
HAIL CAME DOWN THE CHIMNEY YESTERDAY.

:I hav five magits and they are all pyoopas
and they are brown they are not rigglai but
my caterpiler is rigglai.
I HAVE FIVE MAGGOTS AND THEY ARE ALL PUPAE
AND THEY ARE BROWN THEY ARE NOT WRIGGLY BUT
MY CATERPILLAR IS WRIGGLY .

Figure 8.4

A further large domain was also investigated. This consisted of 89 different words from a paragraph in the magazine New Scientist. It was found that with this number of words the time taken for correcting errors in spelling (or merely recognizing the words) grew very long. The few words that were presented were correctly recognized, but the trial was abandoned before completion. There were no special concessions made in the system to reflect the domain of words. If more specialized knowledge had been used, the results could have been obtained much more quickly. In the vision domain it is unlikely that as large a number of objects would become candidate interpretations for scene fragments. The extra two dimensions make relations less likely to succeed, and so the number of interpretations and size of the scene graph do not grow so fast, allowing the solution to be obtained much more quickly.

8.5 DISCUSSION

The success of the spelling correction has shown the power of the system, particularly when the words are not maliciously chosen, and with errors of the kind made by people. The failures and partial failures illustrate some of the shortcomings of the approach. Some of these are artifacts of the particular implementation.

Spelling correction is performed by finding the most likely interpretations for a string of letters, these usually being the correct words. Should there be a large number of similar words, however, no one interpretation may dominate. Often it would be possible to reduce the number of alternatives by using a knowledge of word structure. Most of this knowledge is involved with capturing the fact that strings of letters are one-dimensional. For example,

powerful constraints would be provided if it were known that letters of a word all occur together, without letters of other words interspersed (as opposed to the situation in the three-dimensional vision world).

The system might also make errors because it has no knowledge of the number of letters in a word. When dealing with vision, the number of surfaces in an object was not so important. It would be impossible to see all the surfaces of an object at once. In any case, the number of surfaces in a scene was usually fairly small and the extra dimensions made chance relationships less likely.

In spelling correction, however, and especially when trying to separate words that have been run together, it would be useful to be able to prevent more letters being added to an instance of a word than that word allows. It should also be more difficult to remove letters from a word instance that was completely satisfied. The use of this sort of knowledge would require some alteration to the filtering processs, but would enhance the power of the spelling corrector.

Another result of the one-dimensional nature of words is the much higher incidence of "coincidental" successes in relations than in the vision domain. A large number of hypotheses may be made about what word a particular letter belongs to. Since all hypotheses are made before any filtering, the system can become clogged up. There is usually a lot of circumstantial evidence for many different words (i.e. many relations have succeeded), and it is not clear until all the evidence is available which words have the lesser claims.

Earlier spelling correction systems were based on more statistical methods than the present system. Some authors (e.g.

Blair (1960), Davidson (1962)) derived rules to abbreviate words before matching them with a dictionary. The dictionary contained words and their abbreviations, and a misspelled word was first abbreviated, and then matched letter by letter with the abbreviations in the dictionary. THe method depended strongly on the way on which letters were abbreviated. It was used in restricted domains. For example, Davidson's system operated in the domain of passenger names in an airline reservation system.

A more interesting method depended on the expected frequency with which pairs of letters occur next to each other in written English (Cornew (1968), Omond (1977)). A table of the expected frequencies was required for the method. The algorithm involved first looking up a word in a dictionary of correct words. If no match was found, a misspelling was assumed, and the system searched for the letter most likely to be in error. It did this by examining all pairs of letters in the word. Each letter occurred in two pairs, the first and last letters being paired with a space in addition to the following (preceding) letter. The score of a letter was calculated as the product of the two expected frequencies of co-occurrence for the two pairs in which the letter occurred. These frequencies were obtained from the table. The letter in the word that had the lowest score was assumed to be in error.

The letter was replaced by another letter, chosen so as to maximize the expected pairing frequencies at that position in the word. The word was then rematched with the dictionary. A failure resulted in the process being repeated, first with the next best letters, in order, and then with the next most likely position for an error in the word. After failure with all letters in all positions,

deletion of a letter was tried, followed by insertion. The method was not able to handle transposition of letters.

The method required a great deal of computation, and could handle only single letter errors. Looking at the method from the position adopted in this thesis, it can be seen to be applying adjacency criteria to letters in the words and using the results to determine where the error lies. In this respect it is a less general version of the present work, where relations are more general than adjacency. The earlier discussion of the inadequacy of the PRECEDES relation by itself illustrates a souce of difficulty for the method.

The present system is much more powerful than earlier systems. It can deal with multiple errors in words, and with errors of a more general kind. It is able to correct errors caused by missing letters, added letters, transposed letters, and incorrect letters. In adition, it can separate words run together without spaces and can deal with anagrams and ambiguous letters. A system designed specially for the domain could index words based on more sophisticated information than single letters, and could be made much more efficient.

A rather different method (proposed by H. B. Savin) was reported by Alberga (1965). It involved encoding the pronunciation of each word in the dictionary, and attempting to identify misspelled words on the basis of phonetic similarity to words in the dictionary. This seems a good idea, but Alberga tested it and found it not very effective. It depends heavily on the rules for pronunciation chosen to perform the matching.

The present system can be said to understand the structure of words, but not their meaning. (It could be said to understand the

meaning of specific words like "cube" or "cylinder" when it has models for the objects). A really sophisticated spelling corrector would need more semantic information, and a knowledge of the structure of language. In principle it would be possible to provide such information with another level of models, describing sentences. Such sentences would accept the output of the current system, and correct that where necessary. It would be necessary to deal with the problems of natural language understanding to get the most out of a system with these capabilities.

One of the major advantages of this domain is the ease of acquisition of primitives, and the accuracy with which they can be described. Relations, too, are easily verified, and there is usually no uncertainty as to their result. This should be compared with the situation in vision, where relations need some sort of thresholding in their verification. Because of these advantages, it has proved easier to discover the limitations of the recognition system.

There are other potential uses for word models, such as simple translation, or providing definitions of words, or other messages. This involves changing the information that is associated with the name of the word model, and which is printed out when the word is recognized.

9.0 CONCLUSIONS

This chapter summarizes the achievements of the thesis and points out some of the problems that still remain. It also indicates some possible extensions.

There are two main achievements of this thesis. It has presented a new representation based on a more general notion of a model than is common in vision. It has also shown how the representation can be used in an efficient scene analysis system.

9.1 REPRESENTATION

The representation that was developed dispenses with an implicit requirement for an object model. This requirement is that the number of parts used in the description of an object should be the same as the number of parts in the object. Further, instead of having separate structures for each model, all objects are described in one network, with a single description for all similar parts in all objects. The representation is made viable by labelling each part of the network with the names of the models it described, and by a comprehensive system of indexing.

The advantages of this form of representation are as follows. The representation is compact because nodes are shared by parts of several objects. This allows the space required to store models to

be reduced, and has valuable implications for recognition. Models can be constructed easily from visual data, and new models can be integrated into the network without affecting those models already in existence.

Because of the sharing in the structure, all models that are described by a part of the structure are made available simultaneously when that part of the structure is accessed. Instead of matching with each model individually, all of them are available at once and the work that is done to discriminate between models can be tailored by the context. This allows less work to be done when a scene is analyzed.

The representation is flexible. It was shown that the domain to which the system was applied could be easily changed from three-dimensional vision to one-dimensional spelling correction without changing the representation. It is also possible to describe objects within a domain in different terms, though this requires a means of comparing descriptions when different interpretations have to be evaluated during recognition.

A problem with the representation is one common to all representations. It is the problem of choosing the basic primitives and relations that are used to describe the objects. This choice differs for different domains. In this thesis, it has been allowed to differ also for different objects within a domain.

Although it was possible to give some guidance on how to decide whether or not a model that had already been constructed was adequate, the central problem of choosing the particular primitives and relations in the first place has not been solved. It was stated earlier that it would be useful to have some kind of metric on

relations in order to be able to rank their effectiveness. This would enable models to be constructed using the most useful relations. It would also allow the recognition system to compare the contributions of different relations to an interpretation in a principled way. The ranking of a relation depends on the domain in which it is applied, and it is necessary to treat each case on its merits. Relative weights for a relation were produced automatically by making the weight of a relation inversely proportional to the number of models in which it occurs. This gave an indication of how well the relation distinguished between objects, but assumed that all objects were equally likely to appear in a scene.

Another difficulty concerned the description of surfaces in the implementation. Two numbers are not sufficient to describe the shape of a surface if the surface is to be identified exactly. Finer detail about the shape of a surface is needed, such as whether it is rectangular or triangular. It is also desirable to know more about its outline. All the current description gives is an idea of the approximate greatest extent in two orthogonal directions. It is relatively easy to extract edge information from a range map (see, for example Sugihara (1977) for an elegant approach), and perhaps some information of this sort should have been added to the description. However, the purpose of the descriptions was not to describe surfaces exactly, but to index a set of models that could provide interpretations of the surfaces. The complexity of the surface descriptions affects the number of models that may serve as interpretations for the surface. If the description is very detailed, a lot of work has to be done to form the description, but there may be only a single model that matches the description. If

the description is not detailed, it is easier to construct but may match with several models. If the description is too detailed the recognition problem is effectively being solved at the level of surface description. In that case, a uniform set of tests has to be applied to all surfaces before indexing the model. This paradigm was earlier shown to have disadvantages over the present method. A good description for surfaces should provide a small set of alternative interpretations. This set of interpretations may then be examined to find which interpretation is best.

There are problems with using more complex descriptions than those used in this work. The apparent shape of a surface may be affected by occlusion or shadowing when it appears in a scene. If the description depends on seeing the whole surface, it will not be very useful for recognition without a substantial amount of preprocessing of the scene. Some middle path between the description used in this work and a complete description of a surface seems to be called for, but it is not clear exactly what form this description should take.

The kinds of objects that were modelled by the vision system were only moderately complex. The car, for example, comprised seven surfaces and their relationships. More complex objects were not used largely because of the constraint that models should be constructed automatically from visual data. In the spelling domain, the longest words were made up of about fifteen letters, although the system imposed no limits on the size of its models. The one-dimensional nature of words ensured that the models did not become too complex. It can be speculated that the class of objects that could be effectively modelled without the constraints of automatic learning

and restricted primitive and relation descriptions would be much larger. Any objects that can be described in terms of parts and relations between them can, in principle, be modelled using the representation.

There is an interesting possibility of automatically generalizing models. Suppose there exist models of a cube and of a rectangular parallelepiped, and it is desired to form a model of a more general object to describe both of them. The primitives and relations that refer to each model could be found, and a kind of intersection could be performed on them. For instance, the primitives of the generalized model should have those properties that are common to the primitives in both the cube and the parallelepiped. That is, they should be planar, but of unspecified dimensions. Relations, too, can be intersected. Those that are common to both models can be retained (e.g. adjacency and angle relations), while those that are not common to both models, such as distance relations, can be discarded. The resulting model will be able to describe all the objects whose models it is constructed from, in less detail than those models. Thus it might be possible to produce models for classes of objects from instances of objects in the class. This approach is similar to that used by Winston (1975) but does not rely on a prespecified hierarchy of object types.

9.2 RECOGNITION

The representation encouraged the development of a recognition algorithm that was able to use knowledge about objects to direct the recognition process. The recognition involved first creating a structure to describe a scene, guided by the graph of models. This

structure, or scene graph, initially contained all consistent interpretations for each part of the scene. A second stage of processing examined the scene graph to find those interpretations that were best justified. These were the interpretations that were assumed to describe the scene.

From the very beginning, knowledge obtained from the models was used to direct the processing. Parts of the scene were matched with primitives in the graph of models. These matches initially gave the set of possible interpretations for each surface. In addition, the matches gave access to a set of tests to apply to discriminate between interpretations. These tests would be different for different interpretations, so that it was possible to tailor the discrimination tests for each interpretation. Instead of having to apply a uniform set of tests, some of which would be unnecessary, it was possible to use just those that were relevant. This was especially important because of the possibility that different objects had been modelled in terms of different relations, so that a uniform set of tests would be inapplicable.

Another advantage of the representation was its ability to handle symmetric objects without extra work. Instead of constructing many correspondences between the scene and the model, a single correspondence sufficed to represent all those that were identical up to symmetry.

There is one major problem in using the recognizer. It is the ubiquitous problem of control. There are two processes involved in recognition. One is concerned with finding interpretations for parts of a scene, and the other is concerned with comparing different interpretations and finding the best. As the system was implemented,

these were separate operations. That is, all interpretations were found before any were compared. This sometimes led to large structures being built, most of which were pruned drastically later. It would be better if the pruning could be integrated with the process of finding interpretations. This would keep the structure smaller and mean that unlikely interpretations could be ignored.

One way of approaching this ideal is to have different levels of models. At the moment, all models are object-specific. That is, they describe a single object without reference to the domain. Benefits could be gained by having a model of the domain as well. For example, in the domain of spelling correction, it would help to know that letters of one word could not be interspersed with letters of another word. On an even higher level, words could be recognized more effectively if their part of speech were known. For example, if it were known that a verb were expected at some position in a sentence, then only models of words that were verbs need be considered as interpretations for the letters in that position. Such an analysis would lead inevitably to the problems of natural language understanding.

Domain-specific knowledge could help to decide when it was safe to perform constraint analysis in between finding interpretations. Knowledge about the volume of space occupied by an object, together with the constraint that two objects cannot occupy the same space at the same time, could be used to determine when no more interactions could take place between one part of the scene and the rest. When this was known, that part of the scene could be fully interpreted before more matches were found with other parts of the scene.

It is not clear how this sort of information could be acquired

other than from the user. If it were available, it could be used without greatly altering the recognition algorithm.

9.3 CONCLUSION

This thesis has developed a representation scheme that involves a novel relationship between an object and its model. This leads to a number of advantages. The representation is compact because parts of the structure are shared by several objects. The representation allows models to be constructed easily from visual data, and is general enough to allow the representation of objects from different domains simultaneously. Thus models both of words for the spelling domain and objects for the visual domain can share the same representation structure.

The recognition algorithm that was developed also has a number of advantages, These are gained by using the special features of the representation. The recognition algorithm uses knowledge obtained from the scene to direct the scene analysis. It finds all interpretations for parts of a scene in parallel, and is able to tailor its tests to discriminate between interpretations dynamically according to the scene it is analyzing. There is no problem with symmetric objects because the representation removes the ambiguity in matching.

In practice, the modelling and recognition systems that were implemented proved successful in both chosen domains, thus illustrating their versatility. The simplification of both representation and recognition is a direct result of the generalized concept of model.

BIBLIOGRAPHY

M. R. Adler (1976): Recognition of PEANUTS Cartoons. Proc. 2nd AISB Conference, Edinburgh. pp 1-13.

G. J. Agin and T. O. Binford (1973): Computer Description of Curved Objects. Proc 3rd IJCAI, Stanford, pp629-640.

C. N. Alberga (1965): String Similarity and Misspellings. Comm ACM Vol 10, No 5. pp 302-313.

A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone (1975): A Versatile System for Computer-Controlled Assembly. Artificial Intelligence Vol 6 No 2, pp129-156.

H. G. Barrow, A. P. Ambler, and R. M. Burstall (1972): Some Techniques for Recognizing Structure in Pictures. In:Frontiers of Pattern Recognition. Ed Watanabe. New York, Academic Press.

H. G. Barrow and R. M. Burstall (1974): Subgraph Isomorphism, Matching Relational Structures and Maximal Cliques. DAI Working Paper No 5. Department of Artificial Intelligence, University of Edinburgh.

H. G. Barrow and R. J. Popplestone (1971): Relational Descriptions in Picture Processing. Machine Intelligence 6. Eds B. Meltzer and D. Michie. Edinburgh University Press, pp377-396.

H. G. Barrow and J. M. Tenenbaum (1976): MSYS: A System for Reasoning About Scenes. SRI Artificial Intelligence Center, Technical Note 121.

C. R. Blair (1960): A Program for Correcting Spelling Errors. Information and Control Vol 3, pp60-67.

R. Bornat (1976): Reasoning About Handprinted Fortran Programs. Proc 2nd AISB Conference, Edinburgh. pp 38-46.

J. M. Brady and B. J. Wielinga (1979): Reading the Writing on the Wall. In: Computer Vision Systems, Eds A. Hanson and E. M. Riseman, New York, Academic Press.

L. Carroll (1960): The Annotated Alice. Ed M. Gardner. Penguin Books.

M. B. Clowes (1971): On Seeing Things. Artificial Intelligence Vol 2 No 1. pp 79-112.

R. W. Cornew (1968): A Statistical Method of Spelling Correction. Information and Control Vol 12.

F. Damerau (1964): A Technique for Computer Detection and Correction of Spelling errors. Comm ACM Vol 7, No 3, pp171-176.

L. Davidson (1962): Retrieval of Misspelled Names in an Airlines Passenger Record System. Comm ACM Vol %, No 3, pp169-171.

G. Falk (1972): Interpretation of Line Data as a Three-Dimensional Scene. Artificial Intelligence Vol 3 No 2. pp 101-144.

G. R. Grape (1973): Model Based (Intermediate Level) Computer Vision. Ph.D Thesis, Department of Computer Science, Stanford University.

A. Guzman (1968): Computer Recognition of Three-Dimensional Objects in a Visual Scene. Ph.D Thesis, Project MAC Report MAC-TR-37, Department of Electrical Engineering, MIT.

D. A. Huffman (1971): Impossible Objects as Nonsense Sentences. Machine Intelligence 6. Eds B. Meltzer and D. Michie. Edinburgh University Press. pp 295-323.

B. J. Kuipers (1975): A Frame for Frames: Representing Knowledge for Recognition. AI Memo 322. MIT AI-Lab.

A. K. Mackworth (1977): Consistency in Networks of Relations. Artificial Intelligence Vol 8 No 1. pp 99-131.

D. Marr (1975): Early Processing of Visual Information. AI Memo 340, MIT AI-Lab.

D. Marr (1977): Representing Visual Information. AIM 415 MIT AI-Lab.

D. Marr and N. K. Nishihara (1977): Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. AIM 416 MIT AI-Lab.

M. Minsky (1975): A Framework for Representing Knowledge. In: The Psychology of Computer Vision. Ed P. H. Winston. New York, McGraw-Hill.

R. Nevatia and T. O. Binford (1973): Structured Descriptions of Computer Objects. Proc 3rd IJCAI, Stanford, pp641-647.

R. Nevatia and T. O. Binford (1977): Description and Recognition of Curved Objects. Artificial Intelligence Vol 8 No 1. pp 77-98.

R. A. Omond (1977): A Spelling Correction System (the Speller). Undergraduate Project Report, Department of Artificial Intelligence, University of Edinburgh.

R. Orban (1970): Removing Shadows in a Scene. AI Memo 192, MIT AI Lab.

R. J. Popplestone (1977): A language for Specifying Robot Assembly. DAI Research Report No 29, Department of Artificial Intelligence, University of Edinburgh.

R. J. Popplestone, A. P. Ambler, and I. Bellos (1978): RAPT: A Language for Describing Assemblies. The Industrial Robot, Vol 5, No 3.

R. J. Popplestone and A. P. Ambler (1977): Forming Body Models From Range Data. DAI Research Report No 46, Department of Artificial Intelligence, University of Edinburgh.

R. J. Popplestone, C. M. Brown, A. P. Ambler, and G. F. Crawford (1975): Forming Models of Plane- and Cylinder-Faceted Bodies from Light Stripes. Proc 4th IJCAI, Tblisi, pp664-668.

L. G. Roberts (1965): Machine Perception of Three-Dimensional Solids. Optical and Electro-optical Information Processing (ed. Tippet et al.) pp159-197, MIT Press, Cambridge, Mass.

A. Rosenfeld, R. A. Hummel, and S. W. Zukker (1976): Scene Labeling by Relaxation Operations. IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-6, No 6.

Y. Shirai (1975): Analyzing Intensity Arrays Using Knowledge About Scenes. In: The Psychology of Computer Vision. Ed P. H. Winston. New York, McGraw-Hill.

K. Sugihara (1977): Dictionary-Guided Scene Analysis Based on Depth Information. In: Progress Report on Three-Dimensional Object Recognition. PIPS-R-No 13. Electrotechnical Laboratory, Tokyo.

K. J. Turner (1974): Computer Perception of Curved Objects Using a Television Camera. Ph.D Thesis, University of Edinburgh.

S. A. Underwood and C. L. Coates (1972): Visual Learning and Recognition By Computer. TR 123, Information Systems Research Laboratory, University of Texas at Austin.

D. Waltz (1975): Understanding Line-Drawings of Scenes with Shadows. In: The Psychology of Computer Vision, Ed: P. H. Winston. New York, McGraw-Hill.

P. H. Winston (1975): Learning Structural Descriptions From Examples. In: The Psychology of Computer Vision. Ed: P. H. Winston. New York, McGraw-Hill.

Y. Yakimovsky and J. A. Feldman (1974): Decision Theory and Artificial Intelligence: I. A Semantics-Based Decision Theory Region Analyzer. Artificial Intelligence Vol 5, pp349-371.